EternalBlue CVE-2017-0144
MICS-204 Report 1, Summer 2024

Karl-Johan Westhoff
email kjwesthoff@berkeley.edu

UC Berkleley School of Information
MICS Course 204 Summer 2024

Introduction

EternalBlue [1, CVE-2017-0144] vulnerability has all the traits of a superstar hack, it was probably discovered and developed into a tool by NSA, found and leaked by hackers and quickly thereafter used in some of the worst and most destructive malware attacks the world has seen.

EternalBlue first appeared in a leak[2] from a hacker group named "The Shadow Brokers"[3] (TSB) who published a bunch of zero day exploits they claimed to have stolen from the NSA's "Equation Group [1]"[5] (None of this has been confirmed).

EternalBlue enables the attacker to launch a shell on the target with windows system privileges.

As the vulnerability is in the Microsoft Server Message Block (SMB) protocol, which enables file shares across different users, locations and networks (When you share a folder or access a networks drive, SMB is used) it is not contained by network architecture such as subnets but s able to roam free based on how SMB is managed typically on a corporate network. EternalBlue, giving an attacker a system shell [2], in combination with credentials stealing malware e.g Mimikatz, gives the possibility for some highly potent malware to be crafted, which indeed happened (WannaCry, NotPetya etc.). This leaves EternalBlue as the most expensive software vulnerability - yet.

Exploits

The vulnerability was investigated from by Zian et.al. in [6]. Furthermore "zerosum0x0" did a presentation on how EternalBlue works at DefCon26 [7] and in [8] "h3xduck" did a walkthough (All of which i found useful)

EternalBlue is a consequence of three features/bugs in version 1 of the SMB protocol (SMBv1) [9] :

1. **Buffer Overflow** caused by a typecasting error when casting requests from different Windows filesystems
2. **Unchecked packages** when sending secondary messages memory is allocated based on sender (attacker) controlled offset allowing allocation of memory
3. **"Heap Spraying"** is possible: Data can be written to specific places in memory allowing for code to be inserted and executed

SMB runs on TCP port 445, i.e. the exploit works on a port that needs to be open on the firewall (if you want windows file shares to work). When sending files over the network the SMB protocol uses transactions, where information such as packet size, purpose (the protocol has "special features" such as directory searching etc.) and a list of "File Extended Attributes" (FEAlist) is exchanged in a handshake (SMB Primary Request/Interim Response in Figure 1).
SMB allows for transferring large files where the transaction is split up requiring secondary packets to be send (SMB Transaction Sec. Request 1-N in Figure 1). Large file transfers make it easier to hide suspicious packets.
The FEAlist and secondary packets are important for the exploit.

How it Works

Buffer Overflow The FEAlist is a list of key/value properties which are sent with the transactions.

- The client calculates the size of the FEAlist part of

[1] Nickname for NSA's Tailored Access division (TAO). Darknet Diaries [4] has an excellent episode on the shadowBrokers leak

[2] The DoublePulsar tool, also leaked in the shadowBrokers release is a part of EternalBlue and runs 'exec' in 'kernel mode' on the target

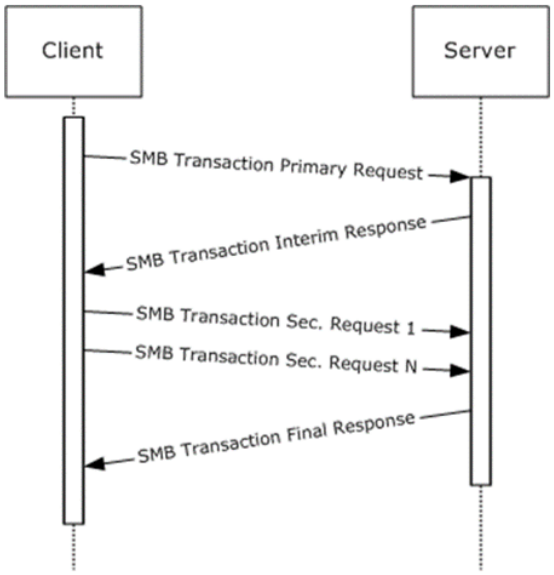

Figure 1: *Request response, SMB protocol. Source: windows learning*

the transaction and sends it to the server with the transaction.

- The server allocates space in memory for the FEAlist, using pointer offsets based on the size of the list (it does not keep a 'list' variable)
- The size of the list is provided by the client, and checked by the server and re-calculated if the size dos not match. However,
  - There are multiple versions of the FEAlist format depending on the filesystem used (OS/2 vs NT). And there is a bug in the conversion of FEAlist file size between OS/2 and NT (bug in a function called "SrvOs2FeaListSizeToNt" in srv.sys [8]) where only half the memory is allocated when sending a OS/2 formatted FEAlist - a 16 byte 'int' is cast to a 8 byte 'short' resulting in a buffer overflow see Ghidra analysis of code from [8] in Appendix Figure 2.

Secondary Transactions, Heap Grooming When the transaction size is large, SMB splits it into subsequent transactions, see Figure 1. The client provides the packet size, which is used as offset for memory allocation i.e. the client (attacker) decides where data is stored in memory. The packets of data are kept as "nonpaged" by srv.sys on the heap until all the secondary packets have arrived, this is indicated with a flag in the last packet (again the attacker controls that).

In Figure 1 'Transaction Sec.' requests can be submitted until the 'N'th' packet containing the end flag. The system does not check the format of the packets, there are 2 different kinds: 'COM' and 'NT' each with their own secondary packets, the protocol assumes the packet format to be that of the final packet send, therefore it is possible to mix the OS and NT packets. And send the first packet as 'NT' with the incorrect FEAList size followed by a subsequent 'OS' type packets. This allows for sending secondary OS packages with the FEAList buffer overflow 'feature' (if we were sending OS only, the double 16bit FEAList would not be allowed)
In appendix a Wireshark dump recorded during exploit is shown. The transaction starts with NT [3] and subsequent transactions are of OS type [4].

Placement of code - From Remote The last thing needed is to place a payload where functions are executed on the stack. Windows has the "Windows Hardware Abstraction Layer (HAL)" which runs in kernel mode and has a static address [8] which can be exploited to execute code.
In between Sec. Request 1-N in Figure 1 a new handshake for a new transaction can be 'weaved in' [5] while keeping the stream of secondary packets flowing (See Appendix Figure 5 Wireshark dump packet No.135 Session "Setup and X process" started before the first transaction FID0X0000 is finished in No.168).

The second transaction is placed in the spots freed, this can be manipulated using the buffer overflows to happen at the address of HAL - now the attackers code runs...

Remediation

Fix the buffer overflow

Patching this was super simple (change the 8bit short to a 16bit int in the "SrvOs2FeaListSizeToNt" in srv.sys, see Appendix Figure 2) - the problem is as always to get the patch rolled out. The patch MS2017-010 was issued March 14, 2017 [10] before EternalBlue was published April 14, 2017 [11] by shadowBrokers (speculation is that NSA nudged Microsoft after being aware that the leak was imminent, they had kept it secret for a long time before that, and the vulnerability showed up before)

[3] "NT Trans Request"
[4] "Trans2 Secondary Request"
[5] "Session Setup AndX"

Randomize HAL's location

Newer version of Windows uses Address Space Layout Randomization (ASLR) for the HAL process, which makes it impossible to predict and control where packages land in memory based on the offsets.

Sandbox processes on the OS

Maybe these application layer interactions with the network should be sandboxed out of the heap so there is no memory cross talk possible.

Conclusion

In hindsight it is obvious to target the SMB protocol as a backdoor. The protocol works close to the operating system (memory is allocated directly on the heap with system privileges), the protocol is designed to be lightweight in terms of package checks and balances (the memory is allocated based on sender specified offsets) Presumably all of this is done to make the transfer of files feel as close to a local operation as possible. Later versions (> 1) of SMB had removed the 'extra' messages used by EternalBlue and were not affected, but at the time systems kept the earlier SMB versions in the stack for backwards compatibility, in fact a recommendation against EternalBlue is to disable version 1 of SMB.

References

[1] *CVE website*. https://nvd.nist.gov/vuln/detail/cve-2017-0144. Accessed: 2024-05-10.

[2] *The Shadow Brokers Leaked Exploits Explained rapid7*. https://www.rapid7.com/blog/post/2017/04/18/the-shadow-brokers-leaked-exploits-faq/. Accessed: 2024-05-12.

[3] *shadowBrokers Wikipedia*. https://en.wikipedia.org/wiki/The_Shadow_Brokers. Accessed: 2024-05-10.

[4] *Darknet Diaries episode 53: Shadoe Brokers Jack Rsyder*. https://darknetdiaries.com/transcript/53/. Accessed: 2024-05-13.

[5] *Kaspersky on the Equation Group kaspersky.com, 2015*. https://www.kaspersky.com/about/press-releases/2015_equation-group-the-crown-creator-of-cyber-espionage. Accessed: 2024-05-10.

[6] Zian Liu et al. "Working Mechanism of Eternalblue and Its Application in Ransomworm". In: *Cyberspace Safety and Security*. Ed. by Xiaofeng Chen, Jian Shen, and Willy Susilo. Cham: Springer International Publishing, 2022, pp. 178–191. ISBN: 978-3-031-18067-5.

[7] *DefCon26 - zerosum0x0 - Demystifying MS17 010 Reverse Engineering the ETERNAL Exploits Def Con 26*. https://youtu.be/9gF3gclIl-c?feature=shared. Accessed: 2024-05-12.

[8] *h3xduck Write up on EternalBlueh3xduck*. https://h3xduck.github.io/vulns/2021/08/22/eternalblue-part8.html. Accessed: 2024-05-13.

[9] *Eternal Blue DoublePulsar Exploit Michael Koczwara*. https://medium.com/dark-roast-security/eternal-blue-doublepulsar-exploit-36b66f3edb44. Accessed: 2024-05-12.

[10] *Microsoft Security Bulletin MS17-010 - CriticalMicrosoft*. https://learn.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010. Accessed: 2024-05-18.

[11] Wikipedia contributors. *EternalBlue — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=EternalBlue&oldid=1219364760.

Appendix

Buffer Overflow Bug in srv.sys

A figure from [8] with code deconstructed from a vulnerable windows srv.sys is shown in Figure 2
Note line 27 the variable "param1" is being casted from an integer (dword) pointer to a short (word). Then in that word the subtraction of itself with another variable piVar1 is stored. Thus we are essentially casting into a short the result of the subtraction of two integers [8].

Decompile: _SrvOs2FeaListSizeToNt@4 - (srv.sys)

```
2  uint _SrvOs2FeaListSizeToNt@4(int *param_1)
3
4  {
5    int *piVar1;
6    int iVar2;
7    int iVar3;
8    int *piVar4;
9    uint local_8;
10
11   local_8 = 0;
12   piVar4 = (int *)(*param_1 + (int)param_1);
13   piVar1 = param_1 + 1;
14   while( true ) {
15     if (piVar4 <= piVar1) {
16       return local_8;
17     }
18     if ((piVar4 <= piVar1 + 1) ||
19        (iVar3 = (uint)*(byte *)((int)piVar1 + 1) + (uint)*(ushort *)((int)piVar1 + 2),
20        piVar4 < (int *)(iVar3 + 1 + (int)(piVar1 + 1)))) break;
21     iVar2 = _RtlULongAdd@12(local_8,iVar3 + 0xcU & 0xfffffffc,&local_8);
22     if (iVar2 < 0) {
23       return 0;
24     }
25     piVar1 = (int *)((int)piVar1 + iVar3 + 5);
26   }
27   *(short *)param_1 = (short)piVar1 - (short)param_1;
28   return local_8;
29 }
```

Figure 2: *The buffer overflow in Srv!SrvOs2FeaToNt, deconstructed using Ghidra, figure from [8]*

Demo using Metasploit

As a demonstration of how easily the EternalBlue vulnerability can be deployed to get a system shell using Metasploit, I ran the exploit on virtual machines locally on my computer:

Attacer machine Kali linux running Metasploit (see Figure 4).

Target machine Early Windows 7 machine (2009) see Figure 3, the exploit is proven to work up to 2017 where a patch was introduced (MS17-010). Only modification is that a folder was shared on the desktop with 'user' permissions (this opens the necessary outbound rules on the firewall)

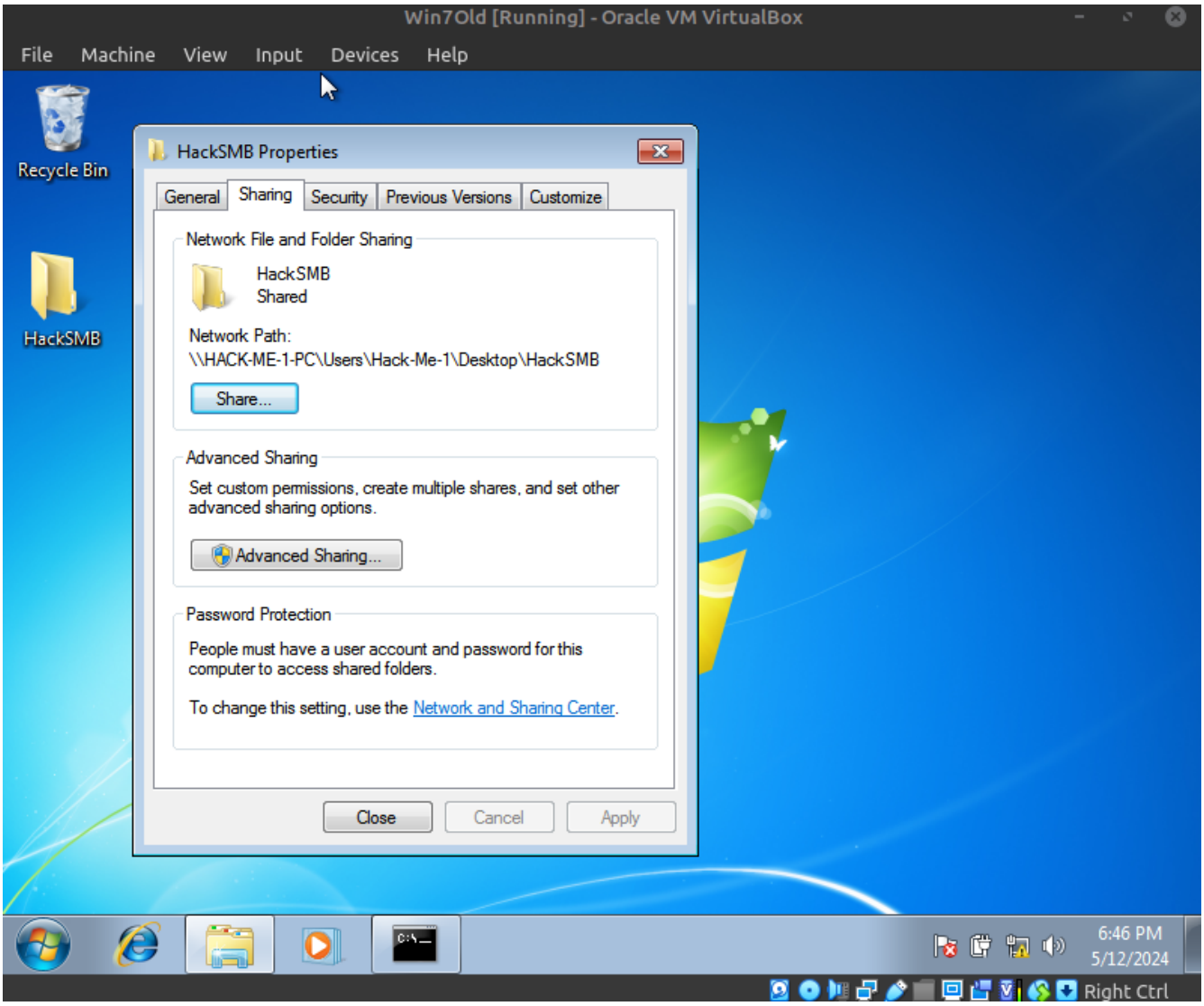

Figure 3: *Exploited windows 7 2009 service pack 1*

Wireshark dump Wireshark dump recorded during exploit is shown in Figure 5, where the transaction starts with NT "NT Trans Request" No.44 and subsequent Transactions are of OS type "Trans2 Secondary Request" No.47-64. In between a new session is started "Session Setup AndX" No.135 and finally the first transaction is terminated in No.168

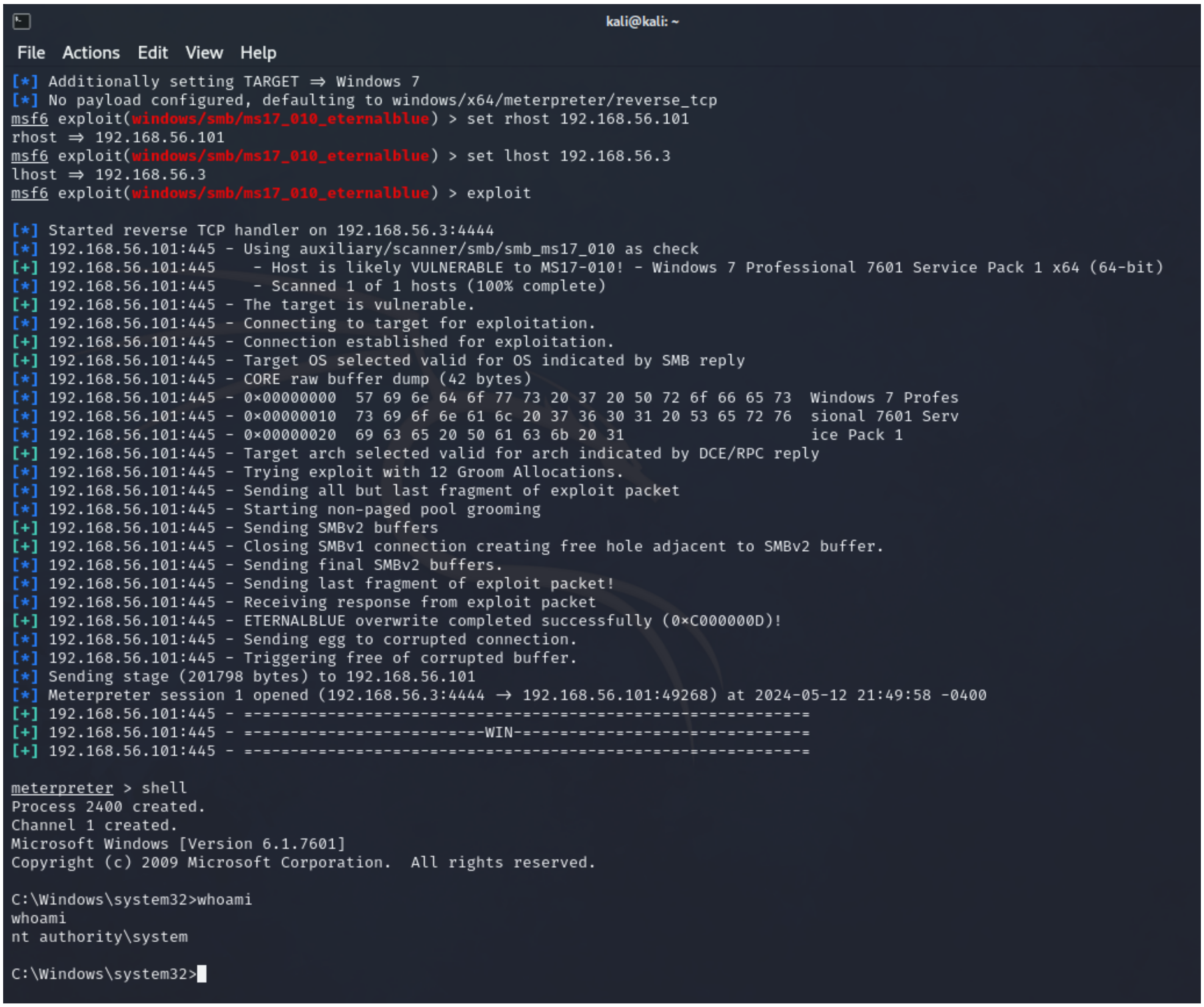

Figure 4: *Very easy with Metasploit*

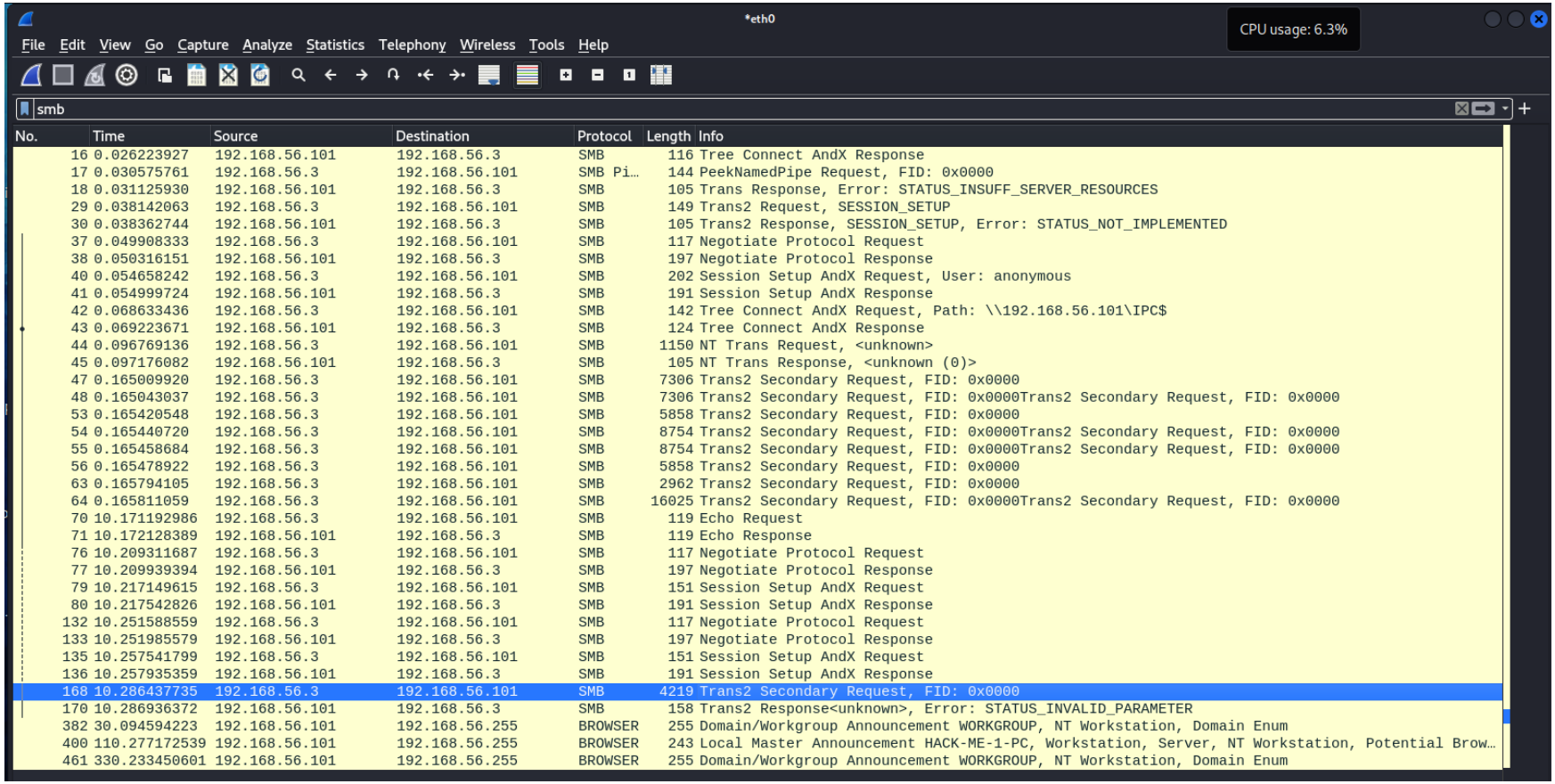

Figure 5: *SMB Traffic during exploit*