PENTERA | Pentera Labs™ Research Series

135 is the new 445 PsExec over Remote Procedure Calls

See how a Pentera security researcher uncovered a new way to use RPC exploits on another port without using SMB.

Yuval Lazar

Table of contents

Purpose

See how a Pentera security researcher uncovered a new way to use RPC exploits on another port without using SMB.

Executive summary

A security researcher's angle on the importance of closely monitoring DCE/RPC and port 135 to prevent lateral movement via PsExec, a favorite Windows Utility. While the attack method outlined in this research works on port 445, it also can be executed on port 135, which is not known.

If we were to nominate a command-line utility in the most useful category, PsExec would be a definitive winner. This tool allows administrators to run remote commands as if they were on the local computer. But unfortunately, not only network administrators love this tool, so do hackers.

Does it apply to my organization?

Of course! If you have Windows computers or a domain.

Who should read this?

Researchers/ethical hackers who want to improve their skills & capabilities or defenders (CISOs) to know they also need to monitor port 135.

Sysinternals 101

Windows Sysinternals is a suite of tools which supplies users with numerous free utilities and resources to manage, diagnose, troubleshoot, and monitor a Microsoft Windows environment. One of these tools is a command-line utility for Windows called PsExec, which was built to replace tools like telnet, that forced you to open up ports and introduce security vulnerabilities. PsExec allows full interactivity for console applications without any setup or installation of a client software, which makes it very easy to use. It can launch interactive command prompts, run as a local system on remote computers, run commands on multiple computers at once and more.

How does it work?

Before we dive in, let's talk for a minute about Remote Procedure Calls. RPC is a protocol that provides high-level communication with the operating system. It relies on the existence of a transport protocol, such as TCP or SMB for carrying the messages between communicating programs. RPC implements a lot of functions that can help a user to create, manage and execute services on the operating system.

PsExec usually uses the SMB protocol to run, and most often runs on port 445 for its common uses. It requires SMB to enumerate the writable shares, so it can use one of the writable shares to upload an executable to it. The tool also uses SMB to supply the user with output of the commands.

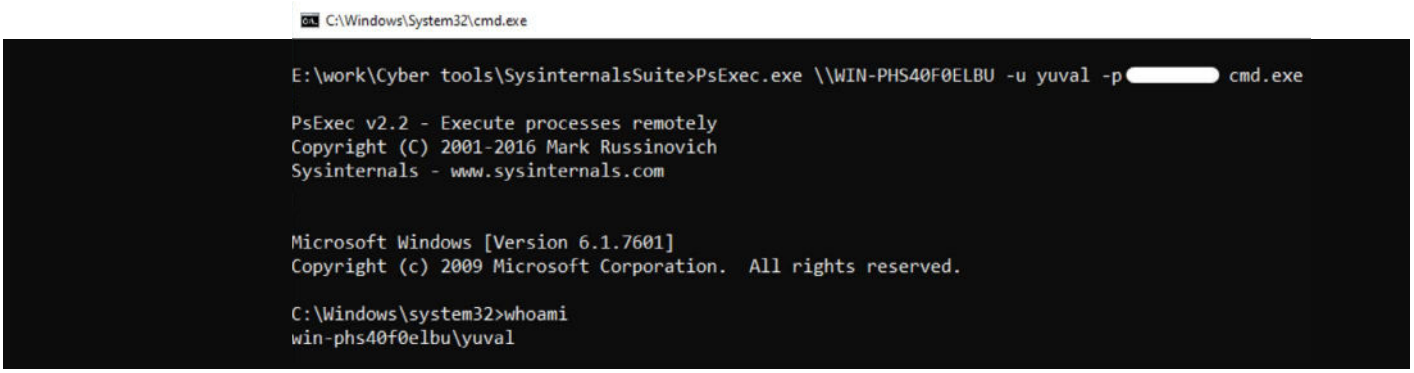
PsExec requires three parameters in order to run - a computer name, credentials, and a command. As we said, it does not require any installations.

All that's necessary for PsExec to work is:

- File and Printer Sharing to be enabled on the remote machine
- An available shared folder

Let's give it a try

Usually, PsExec will require the share folder $Admin, which is an administrative share. Luckily, this is a very common configuration on most Windows computers, so we can get started with running PsExec Using SMB file sharing, the tool will upload an executable called PSEXESVC.exe, then it will use RPC to create and start a Windows service on the remote computer, which in turn will run the program with the relevant arguments.



In the above screenshot, you can see that we are running cmd on a remote computer using PsExec, by supplying a username and password. This is just the tip of the iceberg, and there is much more that can be achieved using this tool.

Can you make it easier?

Since PsExec is such a useful tool, there is another implementation of it in the Impacket Library. Impacket is a collection of Python classes used for working with network protocols, built by SecureAuth Labs. The library provides a set of tools as examples of what can be done within its context, one of which is a python implementation of the PsExec tool.

This implementation also uses an SMB connection and is based on port 445. It uses DCE/RPC methods such as the SVCCTL named pipe, which is used to manage Windows services via the SCM (Service Control Manager).

The basics of this implementation work much the same as the PsExec Windows tool, with just a few tweaks.

The Impacket Library implementation of PsExec has the option to:

- Supply a file to run. If we do not provide a binary file, then the default binary that the program runs is taken from RemCom, another open source project.
- Provide a service name. If not provided, a random 4 letter word is used.

Once the user exits the console or the command has finished, SCM is called to close the service, the exe file is deleted and the SMB connection is disconnected.

Oops, I did it again

So up until now, we talked about two implementations, which operate in the same basic way and use port 445. But what if port 445 is blocked? What can we do then?

Using the Impacket infrastructure, we were able to build an implementation of PsExec based solely on port 135. We found that the SMB protocol is used to upload the binary and to forward the input and output, but as we explained, the commands are executed using DCE/RPC calls, and the processes will run without consideration of the output.

Port	Protocol	Info
135	DCERPC	Bind: call_id: 1, Fragment: Single, 1 context items: EPMv4 V3.0 (32bit NDR)
55922	DCERPC	Bind_ack: call_id: 1, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptan…
135	EPM	Map request, SVCCTL, 32bit NDR
55922	EPM	Map response, SVCCTL, 32bit NDR
49155	DCERPC	Bind: call_id: 1, Fragment: Single, 1 context items: SVCCTL V2.0 (32bit NDR), NTLMSSP_NEGO…
59598	DCERPC	Bind_ack: call_id: 1, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptan…
49155	DCERPC	AUTH3: call_id: 1, Fragment: Single, NTLMSSP_AUTH, User: WIN-PHS40F0ELBU\yuval
49155	SVCCTL	OpenSCManagerW request
59598	SVCCTL	OpenSCManagerW response
49155	SVCCTL	CreateServiceW request
59598	SVCCTL	CreateServiceW response
49155	SVCCTL	StartServiceW request
59598	SVCCTL	StartServiceW response

By using RPC calls we can create a service that will run a command of our choice, and start the service, without the use of port 445. This implementation does not offer a running output, but there are ways to overcome this.

Here you can see an example code from the implementation, that will create a DCERPC connection without the use of SMB as transport or output.

```
def setDCERPCConnection(self, binding):

    rpc = transport.DCERPCTransportFactory(binding)
    rpc.set_credentials(username=self.__username, password=self.__password,
                        domain=self.__domain, nthash=self.__nthash)
    rpc.setRemoteHost(self.remoteHost)
    rpc.setRemoteName(self.remoteName)
    if self.__doKerberos:
        #using kerberos must include using hostname and full FQDN in domain
        rpc.set_kerberos(True, self.__dcip)

    try:
        LOG.info(f"Starting DCERPC connection to {self.remoteHost}")
        rpcsvc = rpc.get_dce_rpc()
        rpcsvc.set_credentials(*rpc.get_credentials())
        rpcsvc.set_auth_type(RPC_C_AUTHN_WINNT)
        rpcsvc.set_auth_level(RPC_C_AUTHN_LEVEL_PKT_PRIVACY)
        if self.__doKerberos:
            rpcsvc.set_auth_type(RPC_C_AUTHN_GSS_NEGOTIATE)
        rpcsvc.connect()
        rpcsvc.bind(scmr.MSRPC_UUID_SCMR)
        LOG.info(f"Successfully connected to {self.remoteHost}/MSRPC_UUID_SCMR")
        return rpcsvc
```

Conclusion

As protectors, we mostly focus on port 445 as the "source of all evil" and monitor it everywhere. Yet sometimes we forget that port 135 can also run SMB, and not only port 445.

Now, SMB is not the only exploitable protocol. DCE/RPC, as discussed above, is a far more valuable protocol for hackers as compared with SMB, yet is many times overlooked and not monitored correctly. DCE/RPC is a primary target used for lateral movement in the network, as is often demonstrated by Pentera's validations - many times without even being noticed by the Blue Team. Don't let this blindspot undermine your own security posture. It is clearly necessary to implement mitigations for port 135 and closely monitor DCE/RPC.

About the author

Yuval Lazar has earned recognition among the Top 25 Women in Cybersecurity for 2020.
After leading research projects in the elite IDF 8200 Unit, Yuval joined Pentera as a Senior Security Researcher. Driven by her love for offensive cyber research, Yuval is constantly searching for new complex attack vectors.

For any questions, feel free to reach out at yuval.lazar@pentera.io

About Pentera

PENTERA

Pentera is the category leader for Automated Security Validation, allowing every organization to test with ease the integrity of all cybersecurity layers, unfolding true, current security exposures at any moment, at any scale. Thousands of security professionals and service providers around the world use Pentera to guide remediation and close security gaps before they are exploited.

For more info, visit: pentera.io

PENTERA | Pentera Labs™ Research Series

135 is the new 445
PsExec over Remote Procedure Calls