< Back to Blog

ZERO NETWORKS LABS

Stopping Lateral Movement via the RPC Firewall

Published **November 27, 2021** by **Sagie Dulce**

TL;DR

The Zero Networks research team built and presented a new tool @BlackHat
called the **RPC Firewall**, a free and open source tool; you can access it here.
Security teams can use the RPC Firewall to detect and protect against
innumerable lateral movement techniques and other forms of remote attacks and
vulnerabilities

Introduction

Over the past few years, we have witnessed (repeatedly) how attackers were able to compromise remote hosts, impersonate privileged accounts and remotely disable security software such as antivirus and EDR to avoid detection. High profile vulnerabilities such as "PrintNightmare" or even "classic" lateral movement techniques via scheduled tasks were utilized by ransomware groups and nation state actors. It seemed that even the best security teams and products are not able to stop an attacker from spreading inside the network.

What is the reason these attacks still manage to spread with ease?

The main reason is that hosts have way too many services exposed over the network, making them susceptible to vulnerabilities and other forms of lateral movement techniques.

This issue is exacerbated when discussing protocols such as RPC, which is prevalent in Windows environments. RPC drastically increases the attack surface available to attackers, by exposing multiple services on the most sensitive servers such as domain controllers. Because administrators don't have granular control on the specific RPC services which are exposed, they end up enabling any RPC traffic; effectively exposing all their RPC services remotely over the network.

We saw the prevalence of RPC not just as a risk, but also as an opportunity, an opportunity to drastically reduce the attack surface by enabling better control of RPC services. To accomplish this goal, we built a tool that can easily protect hosts from unwanted RPC traffic, essentially stopping multiple techniques of lateral movement, reconnaissance, relay attacks and even many vulnerabilities with one simple "trick".

The tool is the RPC Firewall. But before we dive into the details, we need to understand the basics.

What is RPC?

RPC stands for **R**emote **P**rocedure **C**all. It is a protocol that enables a client process to call a remote function hosted in a server process. This server process can reside on the same host, or even on a remote host. When the server process is on a remote host, we call it a remote RPC call. The entire "magic" of RPC is implemented inside the RPCRT4.DLL run time library, which is loaded by the RPC client and RPC server.

https://docs.microsoft.com/en-us/windows/win32/rpc/how-rpc-works
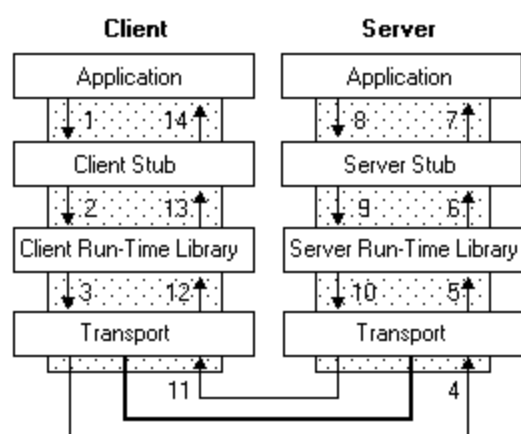
An RPC client can "talk" with a server using a **UUID** (**U**niversal **U**nique **Id**entifier) and an **Endpoint** (the port or named pipe where the server is listening). Once a connection is made between a client and server, the client can call different functions which are identified by their **Opnum** (a numeric identifier which is mapped to the actual function).

To give an example, let's look at the **MS-TSCH** (Task Scheduler Service Remoting Protocol). When browsing through the specification, you will find that there are several UUIDs associated with this protocol, along with a well known endpoint RPC clients can use to communicate with the scheduler service (PIPE⬚tsvc):

The following is a table of well-known UUIDs used in the ATSvc, SASec, and ITaskSchedulerService protocols.

Name	Value	Purpose
GUID_ATSvc	1FF70682-0A51-30E8-076D-740BE8CEE98B	ATSvc UUID version 1.0
GUID_SASec	378E52B0-C0A9-11CF-822D-00AA0051E40F	SASec UUID version 1.0
GUID_ITaskSchedulerService	86D35949-83C9-4044-B424-DB363231FD0C	ITaskSchedulerService UUID version 1.0

The ATSvc and SASec interfaces use the ncacn_np RPC protocol sequence and the well-known endpoint \PIPE\atsvc. The ITaskSchedulerService interface uses the ncacn_ip_tcp RPC protocol sequence and RPC dynamic endpoints.

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-tsch/fbab083e-f79f-4216-af4c-d5104a913d40

One can also find the specific functions referenced by each Opnum:

Methods in RPC Opnum Order

Method	Description
SchRpcHighestVersion	The SchRpcHighestVersion method returns the highest version of the Task Scheduler Remoting Protocol supported by the server. Opnum: 0
SchRpcRegisterTask	The SchRpcRegisterTask method registers a task with the server. Opnum: 1
SchRpcRetrieveTask	The SchRpcRetrieveTask method returns a task definition. Opnum: 2

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-tsch/eb12c947-7e20-4a30-a528-85bc433cec44

This is just the basic concepts of RPC. Of course, there is a lot more to it. If you are interested in learning more, I would recommend the following resources and tools:

- MS-RPCE: official protocol documentation.

- Offensive Windows IPC Internals 2 - RPC: excellent writeup by 0xcsandker.

- RPCView: a must have tool to explore RPC servers.

- NtObjectManager: implements (also) RPC clients via .Net, with a wonderful lecture by @tiraniddo.

- Utilizing RPC Telemetry: guide on how to utilize RPC telemetry to detect lateral movement, by @jaredcatkinson, @v3r5ace and @jsecurity101.

RPC Is Everywhere

Many security professionals don't realize just how common RPC is. RPC communication is happening all the time between various applications and services, both locally and remotely. Any Windows host which is accessible over the network, offers an attacker hundreds, if not thousands, of RPC functions to exploit—either by using stolen credentials or vulnerability.

To give the reader some idea of how prevalent RPC is in attacks, here is a **non-exhaustive** list of some of the attacks and tools that rely on remote RPC:

- Remote Scheduled Task for lateral movement and RCE

- Remote Service Creation for lateral movement and RCE (used by PsExec.exe for example)

- Remote Registry Modification for lateral movement and RCE

- Remote WMI via WMIC.exe for lateral movement and RCE

- Remote DCOM for lateral movement and RCE

- SharpHound for internal reconnaissance

- CornerShot for internal reconnaissance

- PrintNightmare vulnerability which allows RCE

- ZeroLogon vulnerability which allows privilege elevation

- PetitPotam attack which can facilitate various relay attacks

RPC Detection & Protection using Native Capabilities

Ignoring additional applications and services installed on an any given Windows host, there is a limited number of ways an attacker can choose to exploit this host. These are usually RDP, Windows Management and RPC. However, unlike the first two, RPC can be transported on top of multiple protocols, and can be exposed over dynamic endpoints (named pipes or ports), which makes it hard to simply "firewall-off" and forget about it.

Not to mention that the most sensitive servers such as Domain Controllers, must have RPC services open to **any** asset in the network, for the domain to function properly. That way, admins (perhaps unknowingly) expose all their RPC services over the network. This is equivalent to getting through passport control (the firewall) but then having the option to jump on any flight you want.
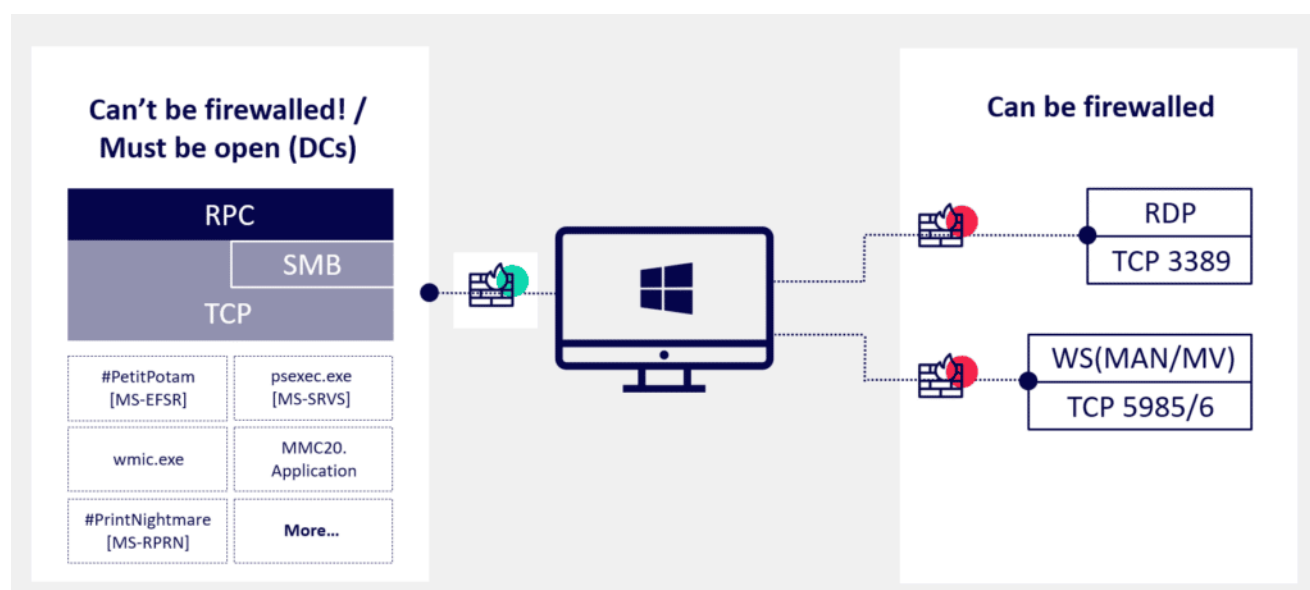
When the "simple" firewall is not an option, what are defenders left with?

The sad news for defenders out there is that Microsoft does not provide enough capabilities that enable security teams to detect and protect against RPC based attacks.

As far a logging goes, the options are not great. Firstly, while there is a documented event for a remote RPC call, it seems that this event is not implemented:

5712(S): A Remote Procedure Call (RPC) was attempted.

09/08/2021 • 2 minutes to read •

It appears that this event never occurs.

Subcategory: Audit RPC Events

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5712

Then there is the ETW option of tracing RPC operations. However, while this produces millions of RPC client and server events per hour, the events also don't contain enough information about the RPC call. For example, you don't know from which remote address the RPC call came, or which user performed it, as can be seen in the following figure:
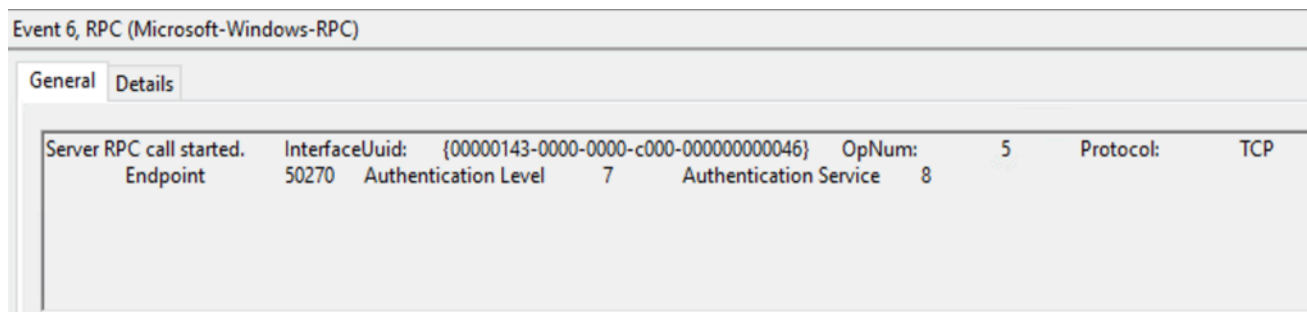
Lastly, the protection offered via RPC filters is not granular enough, and contains some bugs. One example of the lack of granularity is that it is not possible to filter RPC calls based on Opnums. An example of a bug in the runtime is that even though there is a documented condition to filter RPC calls based on remote address in the FWPM_LAYER_RPC_UM layer, in practice this does not work. Microsoft currently says this is an issue they won't fix.

Given all of the issues described, it was clear to us that a new type of solution is required to protect against RPC based attacks. You guessed it, that solution is the RPC Firewall.

RPC Firewall Deep Dive

The RPC firewall is a free and open-source tool, which enables you to audit and block remote RPC calls. The core of the RPC Firewall is implemented in the **rpcFirewall.dll**, which can be injected into processes which are hosting RPC servers.

Once injected, the rpcFirewall.dll first determines if the hosting process is indeed an RPC server. It does so by listing all the UUIDs and endpoints which are registered in the RPC runtime. If there is no registered UUIDs, or if there are no remote endpoints, the rpcFirewall.dll unloads itself.

If the process is verified as an RPC server, the rpcFirewall.dll intercepts any remote RPC call, and based on configuration, decides whether to audit the call and whether to block it. The rpcFirewall.dll extracts the following information from any RPC call:

- **UUID** & **Opnum** of the call

- The server **endpoint** used during for the call

- **Source address** of the client

- **Principal identity** of the client (if the client authenticated)

- The **PID** and **name** of the process hosting the RPC server
  The audit is saved into the Event Log. These logs can then be forwarded to SIEM and used to create baselines, trigger alerts etc. The following is an example of the produced audit:

```
Event 3, RPCFWP

 General   Details

    An RPC server function was called.

    Process Information:
            Process ID:        5656
            Image Path:        C:\Windows\system32\svchost.exe
            RPCRT_Func:        NdrServerCallAll

    Network Information:
            Protocol:          ncacn_ip_tcp
            Endpoint:          53511
            Source Network Address:   192.168.64.1

    RPC Information:
            InterfaceUuid:     86d35949-83c9-4044-b424-db363231fd0c
            OpNum:             0

    Subject:
            Security ID:       OREZ\Administrator

    Detailed Authentication Information:
            Authentication Level:      PKT_PRIVACY
            Authentication Service:    KERBEROS
```

Managing the rpcFirewall.dll is the **rpcFwManager.exe**. This is a simple command line utility, that allows the operator to install/uninstall the firewall and protect specific processes or any RPC server listening for remote RPC calls.

```
install         - configure RPCFWP EventLog & put DLLs in the %SystemRoot%\system32 folder
uninstall       - remove RPCFWP EventLog configuration & remove DLLs from the %SystemRoot%\system32 folder
pid             - Protect specified process ID with RPCFWP <no pid protects ALL processes!>
process         - Protect specified process by name with RPCFWP <no name protects ALL processes!>
unprotect       - Remove RPCFirewall from any protected process
update          - notify the rpcFirewall.dll on configuration changes
```

The RPC Firewall behavior is controlled via a configuration file named **RpcFw.conf**. The configuration is considered line by line. The first matched line with the RPC call parameters determines the firewall outcome. If there is no match, the default

behavior allows any RPC call without auditing it. Whenever changes are made to the configuration file, the firewall can be notified using the "/*update*" command.

To get more information, and to read the code, visit the github repository.

Considerations

- The RPC firewall functions at the application level, so even when RPC traffic is encrypted it still works.

- RPC Firewall is not persistent – the rpcFirewall.dll is not automatically reloaded if a service restarts:
    - It will perhaps be supported in a future version
    - You can DIY, with a simple startup script or scheduled task: RpcFwManager.exe /install RpcFwManager.exe /pid

- Run RPC Firewall with an administrative account – otherwise you cannot protect services

- Walkaround for Protected Processes
    - Protected processes cannot be injected from other non-protected processes, or load an unsigned DLL.
    - For research purposes, use tools such as mimidrv to unprotect processes and research them.
    - In production environment, use rpc filters side by side with RPC Firewall to protect interfaces hosted in protected processes.

Research / Baseline RPC Calls

Whether used in production, or in a test environment, the auditing capabilities of the RPC Firewall are very powerful. They can help you understand the underlying RPC mechanism of RPC attacks, or help you narrow down your RPC attack surface to a minimum.

To audit any RPC call, follow these steps:

1. Install the RPCFW: rpcFwManager.exe /install

2. Update the RpcFw.conf to enable and audit all calls: action:allow audit:true

3. Protect all processes: rpcFwManager.exe /pid

4. Consume the RPCFWP log events and use them to understand and attack / create a baseline

Protect Against RPC Attacks

Once a baseline is created, or a specific attack is identified, this information can be used to create a configuration that is optimized for protection and performance. Meaning, it blocks and audits only unwanted RPC calls, while enabling and not auditing all other calls. That way, security teams can do a one-to-one translation from an RPC Firewall event to an actionable alert.

For example, let's create a configuration that protects our domain controllers against the DCSync attack. We will do so by blocking the underlying RPC protocol used for this attack(MS-DRSR with the UUID e3514235-4b06-11d1-ab04-00c04fc2dcd2), but only when it is not coming from another DC address. That way

the domain can still perform replications, while not allowing it from any other source.

The following configuration demonstrates how to achieve protection from DCSync in a simple environment with two other domain controllers. uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 addr:<dc_addr1> action:allow uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 addr:<dc_addr2> action:allow uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 action:block audit:true action:allow audit:false

Summary

Defending against lateral movement and other remote RPC attacks is almost impossible without the right tools. We hope the RPC Firewall will be an important milestone in mitigating many forms of remote attacks and will assist security teams in defending their environment.

This is an ongoing project. Please don't hesitate to reach out to us and share your experience, thoughts, and issues with the RPC Firewall at support@phpstack-717800-4003432.cloudwaysapps.com.

Subscribe for new posts

Email*

Subscribe

ZERO.
Networks

Get a demo

Skip the small talk, speak to a specialist.

Get a Demo