Sign in

zeronetworks / rpcfirewall   Public

Notifications      Fork 70      Star 480

<> Code      ⊙ Issues      ⑂ Pull requests      ▷ Actions      ⚠ Security      ∿ Insights

master                    Go to file        <> Code ▾

sagiesec  Update RpcFw.conf.Both  •••        3eb9816 · 2 months ago      ⟲

| Configuration_templa... | Update RpcFw.conf.Both | 2 months ago |
| rpcFirewall | Update dllmain.cpp | 6 months ago |
| rpcFwManager | Doing the rpc call sync... | 7 months ago |
| rpcMessages | Adding SID field in RP... | last year |
| .gitignore | feat(rpcFirewall): auditi... | 2 years ago |
| LICENSE | Update License | 3 years ago |
| README.md | Update README.md | 6 months ago |
| RPCFW.sln | feat(config): Add confi... | 2 years ago |

## About

*No description, website, or topics provided.*

📖 Readme
⚖ View license
∿ Activity
▣ Custom properties
☆ 480 stars
◉ 17 watching
⑂ 70 forks

Report repository

## Releases 15

🏷 v2.2.4  Latest
on May 14

+ 14 releases

## Packages

No packages published

📖 README      ⚖ License

release v2.2.4   downloads 1.6k

# I Need More Information

Check out our [RPC Firewall](#) blog post or our [BlackHat talk](#) to gain better understanding of RPC, RPC attacks, and the

## Contributors 7

solution: RPC Firewall.

Join our |Zero| Labs Slack Community workspace for any questions, issues, or simply to shout out.

We would love to hear from you also via email (if you are that type of person). Contact us at support@zeronetworks.com

# Get Started

The following tutorial shows basic installation and setup of RPC Firewall, as well as a demo of how it protects against various RPC-based attacks.
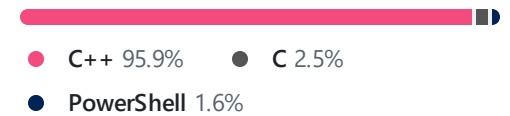


# Why should I care?

RPC is the underlying mechanism which is used for numerous **lateral movement** techniques, **reconnaissance**, **relay** attacks, or simply to **exploit vulnerable RPC services**.

DCSync attack? over RPC. Remote DCOM? over RPC. WMIC? over RPC. SharpHound? over RPC. PetitPotam? over RPC.

## Languages

C++ 95.9%     C 2.5%

PowerShell 1.6%

PsExec? over RPC. ZeroLogon? over RPC... well, you get the idea
:)

# Terminology?

Throughout this document, we will use the following terms:

- **RPC Firewall**: Refers to the actual RpcFirewall.dll, which is
  loaded into various RPC servers in order to protect them.
- **RPC Filters**: Refers to the native Windows RPC filtering
  mechanism.
- **RPCFW Configuration**: Refers to the RpcFw.conf file,
  which is used to control how *RPC Firewall* and *RPC Filters*
  behave.
- **RPCFW Manager**: The command line interface, which
  gives users access to both *RPC Firewall* and *RPC Filters*

# What is it used for?

## Research

Can be used to to **audit** all remote RPC calls. Once executing
any remote attack tools, you will see which RPC UUIDs and
Opnums were called remotely.

See an example configuration [here](#).

## Remote RPC Attacks Detection

When the *RPCFW Configuration* is configured to audit, events
are written to the Windows Event Log. *RPC Filter* events are
written to the security log, with event ID 5712. *RPC Firewall*
logs are written at *Application/RPCFW*.

Users can forward these logs to their SIEM, and use it to create
baselines of remote RPC traffic for various servers. Once an

abnormal RPC call is audited, use it to trigger an alert for your SOC team.

We integrated several Sigma rules which can be used to detect unusual RPC activities and attacks.

## Remote RPC Attacks Prevention

The *RPCFW Configuration* can be configured to block & audit only potentially malicious RPC calls. All other RPC calls are not audited to reduce noise and improve performance.

Once a potentially malicious RPC call is detected, it is blocked and audited. This could be used to alert your SOC team, while keeping your servers protected.

To supplement protection, you can use the RPC Filtering capabilities, which are also supported via the RpcFwManager.exe.

An example of such configuration can be found here.

# What are the RPC Firewall Components?

It is made up from 3 components:

1. **RpcFwManager.exe** - Acts as a standalone command line management tool for RPC Filters and Firewall deployment. Also acts as the installed service for deploying the RPC Firewall.
2. **RpcFirewall.dll** - Injected DLL which performs the audit & filtering of RPC calls.
3. **RpcMessages.dll** - A common library for sharing functions, and logic that writes data into Windows Event Viewer.

# Using RPC Firewall or RPC Filters?

While there are pros and cons for using each method, there are a couple of major benefits for using *RPC Firewall* which require special attention. These are:

1. **Granularity**: The RPC Firewall is applied for each RPC call, and can be used to create more granular controls for specific RPC functions.
2. **Source Address Identification**: *RPC Firewall* is better at determining the source address of the caller. *RPC Filters* cannot identify the source address of calls made over named pipes (SMB), which means **RPC filtering rules will fail for RPC calls made over named pipes!!!**
3. **Bugs!**: RPC Filters suffers from various bugs which Microsoft are not enthusiastic on fixing. A couple of examples are: IP ranges don't work, and you can't apply a "catch all" filter, as it could damadge several Domain Controller services such as NetLogon.

On the other hand, *RPC Filters* are greate for bulk allow or deny of entire UUIDs, as they do so without any issues.

# How to use?

## What's the status?

Prior to running any command, it is recommended to check the status of the deployment. This is done by issuing the '/status' command:

```
RpcFwManager.exe /status
```

This will show the status of both the *RPC Firewall* and the *RPC Filters*. This will output detailed information about the installation status, and also the running status of the deployment.

## The 'fw' / 'flt' suffixes

Almost every command can be suffixed with *fw* or *flt* at the end. This tells *RPCFW Manager* whether the command is applied to *RPC Firewall* ('fw'), *RPC Filters* ('flt') or both when not using any suffix.

## Installing / Uninstalling

Peform installation of the relevant feature (*RPC Filters / RPC FIrewall* / both).

- **RPC Firewall Installation**: Configures the event log, drop relevant dlls into the %SystemRoot%\System32 folder, and configures the **RPCFW** application log for the Event Viewer. It also installs the "RPC Firewall" service, which will be used for persistence.
- **RPC Filters Installation**: Enables the security audit of RPC events. These can be found under the Securit log, with event ID 5712.

*Make sure the event viewer is closed during install/uninstall.* Also, it is good practice to stop & uninstall before installation, to make sure there aren't any previous versions installed.

```
RpcFwManager.exe /stop
RpcFwManager.exe /uninstall
RpcFwManager.exe /install
```

Uninstalling does the opposite. Also here it is good practice to ensure the service is stopped prior to uninstallation.

```
RpcFwManager.exe /stop
RpcFwManager.exe /uninstall
```

# Starting / Stopping

*RPC Filters*, by their nature, are applied system-wide, to any RPC server. Applying such filters is also persistent across reboots. Any new or old process will be protected by *RPC Filters*, according to the *RPCFW Configuration*.

*RPC Firewall* is injected and protects any RPC server process which is listening for remote RPC calls. This is done by injecting *RPC Firewall* into processes which load the RPCRT4.DLL (the RPC Runtime). Once loaded, *RPC Firewall* will detect whether the RPC server is listening for remote RPC calls. If not, it unloads itself. If the process is a valid RPC server, the rpcFirewall starts to audit & monitor incoming RPC calls, according to the *RPCFW Configuration*.

The recommended method to protect RPC services is by using the '*/start*' command. This starts the *RPC Firewall* service (for 'fw' or no suffix), and creates *RPC Filters* (for 'flt' or no suffix).

```
RpcFwManager.exe /start
```

It is also possible to use *RPC Firewall* ad-hoc, to protect specific processes. This will not start the *RPC Firewall* service, and will not persist across reboots.

To protect a single process by Process ID (PID):

```
RpcFwManager.exe /start pid <pid>
```

To protect a single process by name:

```
RpcFwManager.exe /start process <process name>
```

To stop the protection, simply issue the '*/stop*' command with the appropriate suffix (it is not possible to stop protection for a specific process).

```
RpcFwManager.exe /stop
```

According to the suffix used ('fw' or 'flt' or none), this will stop the *RPC Firewall* service and unload the *RPC Firewall* dll from all processes. For *RPC Filters*, it will delete all filters generated by the *RPCFW Manager*.

## Persistency

Once started, the *RPC Firewall* will persiste across reboots. *RPC Filters* are also persistent.

## Configuration

The *RPCFW Manager * looks for a *RpcFw.conf* file, in the same directory of the executable. This file uses the following configuration options:

*Important*: Each each configuration line should be prefixed with either '*fw:*' or '*flt:*' to indicate whether the line is applied for *RPC Firewall* or *RPC Filter*.

| Parameter Name | Explanation | Supported by ... |
|---|---|---|
| opnum: | Match a RPC opnum | RPC Firewall |
| verbose: | Can be either **true** or *false*. When true, outputs debug informaiton for specific RPC calls (default false) | RPC Firewall |

| | | |
|---|---|---|
| prot: | Matches the used protocol to any of the [protocol sequence constants](#) | both RPC Firewall and Filters |
| addr: | Match a remote IP address (IPv4 or IPv6, including CIDR) | RPC Firewall and partially by RPC Filters, [read more here](#) |
| uuid: | Match a specific uuid | both RPC Firewall and Filters |
| action: | Can be either **allow** or **block** (default allow) | both RPC Firewall and Filters |
| audit: | Can be either **true** or *false*. Controls whether events are written to the *RPCFW* log (default false) | both RPC Firewall and Filters |
| sid: | matches an authenticated user to a [security identifier](#). Could be specific user or group. | both RPC Firewall and Filters |

## Seeing More Things

To see more RPC related "things", there is a *show* command. For now, it only shows protected processes, which cannot be protected with the RPC firewall module.

```
RpcFwManager.exe /show
```

The configuration order is important, as the first match determines the outcome of the RPC call.

For example, the following configuration snippet will protect a DC from a DCSync using *RPC Firewall*. It does so by enabling the "dangerous" opnum 3 DRSGetNCChanges of the MS-DRSR UUID only from other domain controllers.

```
fw:uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 add
fw:uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 add
fw:uuid:e3514235-4b06-11d1-ab04-00c04fc2dcd2 opr
```

Whenever the configuration changes, you need to notify the rpcFirewall.dll via the update command:

```
RpcFwManager.exe /update
```

## Viewing Logs

For *RPC Firewall*, open the Event Viewer -> Applications and Services Logs -> RPCFW. Event Ids 1 and 2 refer to 'protect' and 'unprotect' events. Event 3 audits the actual RPC calls. Also, add the Keywords column. This column would contain Audit Success/Failure, implying whether the RPC call was blocked or not.

For *RPC Filters*, open the Event Viewer -> Security-> RPCFW. Filter for event ID 5712.

## Can I Contribute?

Yes! Don't be shy to do a pull request.

## I want RPC Firewall to do more things!