Mitigating Privilege Escalation in Fileset Executables

What

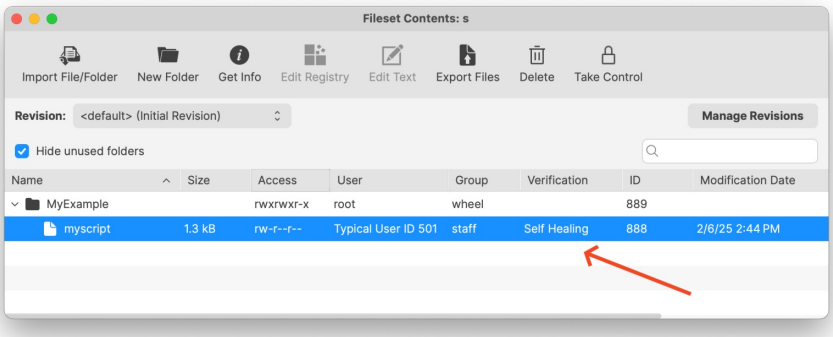
> This article reflects the behavior in FileWave 16.0.0+

When deploying Filesets in FileWave;

- It's possible to include a Verification Script that runs every 24 hours (or at system restart) on Windows and macOS.
- You might also have a Requirements Script that runs every 2 minutes to test for a condition needed to install something.

These scripts are typically used for tasks such as re-applying Group Policy settings—using tools like LGPO.exe on Windows—as an example. However, if an executable (like LGPO.exe) is replaced by a non-administrator, there's a risk that the malicious replacement could be executed with elevated privileges (SYSTEM on Windows or root on macOS).

By default, Filesets benefit from a self-healing mechanism. If an executable is modified, the Fileset will restore the original file before the Verification Script executes. An image below shows this status and picking Get Info for a file may be used to change this on a per-file basis. Properties for a Fileset can be used to change the behavior across the entire Fileset as well.



This same protection applies when the executable is deployed and then called by a Custom Field—the Fileset's verification step precedes the Custom Field execution, thereby mitigating the risk of local privilege escalation.

However, there is one scenario where this protection of Self-Healing does not apply: the Blocker Script, which by default runs every 5 minutes. Its more frequent execution window could allow a local user to replace an executable before the next Fileset verification occurs. However this too can be mitigated as we will discuss below.

When/Why

This article is particularly relevant in environments where:

- Executable files (e.g., LGPO.exe) are deployed via Filesets that also include a Verification Script or are triggered via Custom Fields.
- There is a risk of local users, who lack administrator rights, replacing these executables to attempt privilege escalation.
- The deployment involves Blocker Scripts, which run at more frequent intervals (every 5 minutes), thus presenting a window where an attacker might replace a file before it is healed by the Fileset.

Why is this important?

Ensuring that only the intended executable is run with elevated privileges is critical for system security. A malicious modification could lead to unauthorized privilege escalation. Mitigating this risk by enforcing strict file ownership and permission settings prevents non-administrative users from replacing or modifying executable files.

How

To mitigate this risk, it is recommended to either pick properties on the file in the Fileset and properly securit it or use a Postflight Script in your Fileset deployment that sets the file permissions so that only SYSTEM (on Windows) or root (on macOS) can modify the file.

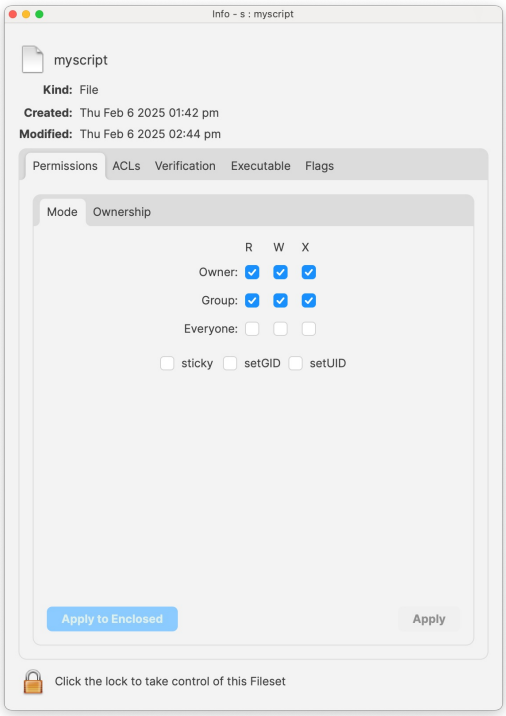
On Windows systems as of FileWave 16.0.0 the c:\ProgramData\FileWave\ directories like FWClient and the scripts directory are restricted so that a non-Administrator is unable to manipulate files located there. This is one way that FileWave will prevent a local attack by default. Files on Windows will also inherit the permissions of the directory that they are deployed in to which will help for a

more consistent permissions experience.

For some examples of setting permissions on files please take a look at the below. For the scripts you could have them run as Fileset scripts when deploying the files to adjust permissions to how you'd like.

macOS Example (Properties of file in Fileset)

In the below images you can see an example file in a Fileset where Get Info has been clicked in the toolbar when viewing the Fileset and the permissions have been adjusted so only root:wheel has access to the file. Make sure you click Apply on each screen to make the setting change.



macOS Example (zsh Script)

Below is an example zsh script that sets the ownership to root:wheel and the permissions to 770 for the target executable.

```
#!/bin/zsh
# Define the path to the executable deployed via the Fileset
TARGET_FILE="/path/to/executable"

# Change the ownership of the file to root:wheel
chown root:wheel "$TARGET_FILE"

# Set the permissions to 770 (owner and group have full permissions, others have none)
chmod 770 "$TARGET_FILE"

# Explanation:
# - chown root:wheel: sets the file owner to root and group to wheel.
# - chmod 770: grants read, write, and execute permissions to the owner and group,
#   while denying any permissions to others.
```

Windows Example (PowerShell Script)

The following PowerShell script sets the file's owner to SYSTEM and adjusts the permissions so that only Administrators and SYSTEM have full control. This ensures that no non-administrator can replace the file.

```
# Define the path to the executable deployed via the Fileset
$TargetFile = "C:\Path\To\executable.exe"

# Set the owner of the file to SYSTEM using icacls
icacls $TargetFile /setowner "SYSTEM"

# Remove inherited permissions to ensure only our defined permissions are in place
icacls $TargetFile /inheritance:r

# Grant full control (F) to SYSTEM
```

```
icacls $TargetFile /grant SYSTEM:(F)

# Grant full control (F) to Administrators
icacls $TargetFile /grant Administrators:(F)

# Explanation:
# - icacls /setowner "SYSTEM": sets the file owner to SYSTEM.
# - icacls /inheritance:r: removes inherited permissions so only our custom permissions apply.
# - icacls /grant SYSTEM:(F) and /grant Administrators:(F):
#   ensures that only SYSTEM and Administrators have full control over the file.
```

Related Content

- FileWave Version 16.0.2 Security Notices
- Verification
- Script Best Practices
- Fileset Scripts Overview

Revision #6
★Created 14 February 2025 17:18:00 by Josh Levitsky
✎Updated 21 April 2025 12:04:47 by Josh Levitsky