



(S) Engineering Development Group

(S) Pandemic
V1.0

(U) Tool Documentation

Rev. 1.0
17 April 2014

Classified By: 2417940
Reason: 1.4(c)
Declassify On: 25X1, 20640418
Derived From: CIA NSCG COL S-06

Contents

Contents.....	2
1 (U) Tool Summary.....	3
2 (U) Release Notes.....	3
3 (U) User's Guide.....	3

(S) Pandemic 1.0

1 (U) Tool Summary

(S//NF) Pandemic is a tool which is run as kernel shellcode to install a file system filter driver. The filter will 'replace' a target file with the given payload file when a remote user accesses the file via SMB (read-only, not write). Pandemic will not 'replace' the target file when the target file is opened on the machine Pandemic is running on. The goal of Pandemic is to be installed on a machine where remote users use SMB to download/execute PE files.

(S//NF) Pandemic does NOT//NOT make any physical changes to the targeted file on disk. The targeted file on the system Pandemic is installed on remains unchanged. Users that are targeted by Pandemic, and use SMB to download the targeted file, will receive the 'replacement' file.

(S//NF) Pandemic can operate against 32 and 64 bit targets, but 1.0 was only developer tested against 64-bit targets due to the CONOPs that was in mind at the time.

2 (U) Release Notes

(S) Version 1.0 is the initial version

3 (U) User's Guide

3.1 (U) Change Log

Table 1: (S) Change log (contents SECRET)

Revision	Date	Author	Notes
1.0	17 April, 2014	AED/RDB	Initial version.

3.2 (U) File Information

Table 2: (S) File information (contents SECRET//NOFORN)

Pandemic_Builder.exe	b0965e3a0f70772a19d98dc3c5ee5f45
Control.dll	221b76333bcac70fc029ffdef0b781b

3.3 (U) File/Registry Access

(S//NF) Pandemic registers a minifilter driver using Windows' Flt* functions. As a result, FltMgr requires that all drivers registering as minifilters contain certain registry keys. Pandemic uses the 'Null' service key (on all Windows systems) as its own driver service key. Pandemic will create 2 sub keys and 3 values under the 'Null' service key in the registry. These values and sub keys are deleted when Pandemic is uninstalled at the

end of its configured run timer, or when it is uninstalled via a special F&F DLL. These keys will NOT//NOT be deleted if the system is rebooted before the aforementioned scenarios occur.

3.4 (U) Configuration

(S//NF) Pandemic comes with a configuration utility which builds a binary file for execution via ShellTerm. Pandemic_Builder has the following arguments:

- --target <string name>: This is a search string to use when watching for the target file to 'replace' on the file server. Pandemic only matches if the string given here is the last part of the file path. For example:
 - --target Winhex\WinHex.exe will match in the following case
 - C:\users\administrator\desktop\winhex\winhex.exe
 - It will not match if the file being opened is:
 - C:\users\administrator\desktop\winhex\winhex.exe.dll
 - Be sure to choose this string carefully, so only the targeted PE is replaced.
- --replace <string path>: Path to the local file that will be used to replace the targeted PE. This should be a file on the configuration machine. This file will be opened, and its contents read and stored in the payload binary. The current limit for the replacement PE is 30 MB
- --timer <32-bit number>: This is the effective time to live (in MINUTES) for Pandemic on target. Once Pandemic has been running for the given amount of minutes, Pandemic will uninstall itself. It can be uninstalled before this time by either a system shutdown, or the Pandemic_Uninstall DLL.
- --delay <32-bit number>: This is the time, in minutes, that Pandemic will wait before it begins replacing the targeted PE with the payload PE. Pandemic can be manually uninstalled in this time period. This time period does not//not count against any --timer value.
- --name <device name string> <link name string>: Two arguments that tell Pandemic how to name the device object and link name for the uninstall device. Pandemic creates a device and device symbolic link for on-demand uninstalling.
 - Ex. --name NTPNP_PCI0046 Scsi3:
- --sids <list of SIDs, space separated>: List of SIDs to target when performing file replacement. Should be in the standard Windows string SID format.
 - Ex. --sids S-1-5-21-708247480-2834978148-2381576337-1001 S-1-5-21-708247480-2834978148-2381576337-1003
 - Current limit is 64 SIDs
- --badsids <list of SIDs, space separated>: List of SIDs to NOT target when performing file replacement. Users with these SIDs will NOT//NOT get the replacement PE (even if no --sids list is given). Blacklist will TRUMP the whitelist. Ex. A SID on --sids will be ignored if it also appears in --badsids
- --logfile <log file path on target>: Path on target to save the log file to. The log will contain a timestamp-SID entry for each read operation performed on the target PE file by a targeted SID.
- --writetime <MM/DD/YYYY HH:MM:SS>: Optional. Value used to set the log's write time value after updating the log. If using --writetime, you must include a

date AND time as shown above, you cannot use partial information. This will also be used as the file's last Changed time.

- --createtime <MM/DD/YYYY HH:MM:SS>: Optional. Value used to set the log's create time value after updating the log.
- --accesstime <MM/DD/YYYY HH:MM:SS>: Optional. Value used to set the log's access time value after updating the log.

(S//NF) Pandemic_Builder will spit out the file pandemic_AMD64.bin (or pandemic_x86.bin for 32-bit version). This file will be used by Shellterm for installation.

(S//NF) *NOTE* Pandemic_Builder does not do any checks on the replacement file beyond that it exists. If the --replace file does not exist, an error will be printed to screen (even though a .bin file is generated). If the file does exist, then the file is used. If you accidentally give a .txt file to replace an EXE, that will cause issues. Make sure you double check which file you're using for replacement.

3.5 (U) Installation and Operation

(S//NF) Pandemic will install via Shellterm's shellcode installer. Included with Pandemic is a sample python script that enables the shellcode installer functionality within Shellterm. This script is not//not required to use Pandemic, but *some* script is required to use Shellterm's shellcode loader functionality.

(S//NF) To use the sample python script, first drop it into Shellterm's script folder. This folder is determined in Shellterm's configuration file. Once installed, attach to an active session on the target, and make sure the .bin file generated earlier is on the Shellterm machine. Run the following command to install Pandemic on target:

```
> kshellcode '<path to .bin file>/pandemic_AMD64.bin'
```

(S//NF) Pandemic can take ~10-15 seconds to install and return back. An error code in the 500-550 range indicates that a required API function could not be loaded. This is an issue that is likely to not resolve itself by re-trying the install, and you should talk to the developer (remember the exact code). Pandemic will return 0 if it was able to kick off the installation thread. *This does not mean that Pandemic was successfully installed*. Shellterm's shellcode launcher specification prevents Pandemic from doing the full install in the initial thread Shellterm provides (could cause Shellterm instability/crash). Therefore, there are several installation steps that happen in a new thread, and can't report error codes back to Shellterm. The following methods are good ways to verify successful installation:

- Through Shellterm, use the reg command to check for the existence of the Pandemic keys/values.
 - Command: reg -q HKLM\SYSTEM\CurrentControlSet\Services\Null
 - You should see a sub-key called 'Instances' Querying that, you should see another sub-key called 'Null'.

- Checking the global objects on the system. Look for a device with the name configured in Pandemic, or a symbolic link with the name configured in Pandemic

3.6 (U) Verification/Uninstall

(S//NF) The DLL Control.dll is included in the delivery. This DLL is a simply Fire and Forget (v2) DLL which can perform two different functions: Checking that Pandemic is installed, and uninstalling Pandemic. The usage information through Shellterm is simply 'loaddll -a "<uninstall string configured above> [-c | -u]" <DLL path>

- -c: Returns successful status (0) if the Pandemic device exists on the target. If the device was not found, the value 0x2 will be returned. The latter indicates that Pandemic either isn't finished setting up (unlikely) or Pandemic is not running.
- -u: Instructs Pandemic to exit as soon as possible. Pandemic will check every 15 seconds to see if it should exit or not (up to any --timer value configured). The time to complete uninstall (when a -c command would return 0x2) could take roughly 15 seconds after issuing a -u command.

(S//NF) The uninstall string needed is the device link name that was used when configuring Pandemic (--name <device name> <**device link name**>). This value should be in your receipt file for the binary. Without this value, you cannot uninstall Pandemic short of guessing the value, figuring out the value (listing the system's objects and guessing), or rebooting the machine. If Pandemic has a --timer value configured, it will exit upon that timer expiring.

3.7 (U) Known issues

(S//NF) The following are known/potential issues that may arise while using Pandemic:

- (S//NF) If the remote user goes to copy the targeted PE on the file server to their local machine, and the remote user has a file with the same name as the targeted PE in the destination folder, Windows will ask the user if they wish to replace the file or cancel the copy operation. The issue is the Windows alert box will report that the new file's size being the size of the targeted PE, not the replacement PE.
 - For example: The targeted file is Pexplorer.exe, size 4.5 MB. The replacement file is NOTEPAD.exe, size 67 KB. If the remote user copies down pexplorer.exe to a local folder with that same file name, Windows will ask the user if they wish to overwrite/cancel the copy. When it does this, it will say that the size of the remote copy is 4.5 MB. However, after the operation is complete, the user will have only downloaded the replacement PE file of size 500 KB.

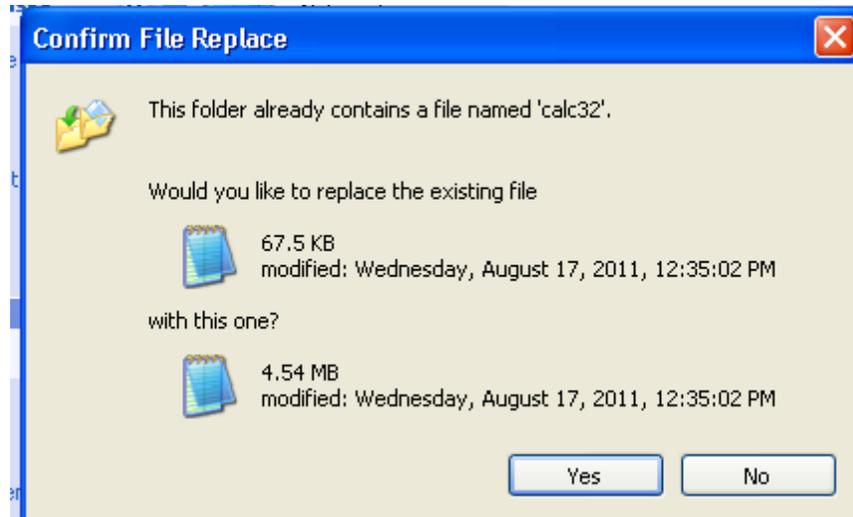


Illustration 1: (S//NF) The same file is copied twice from the remote file share to the user's local disk. As you can see, the file size Windows reports is vastly different, even if the user only gets the smaller replacement file

- (S//NF) If the target file server is running in a VM, the host attempts to drag and drop a folder into the file server VM, and the folder being copied into the file server VM has the targeted path, then VMWare will throw an error. This is likely due to how Windows is classifying VM drag and drop operations on the guest, and Pandemic can't tell if it's a normal SMB operation or a weird VMWare operation.
- (S//NF) If a target user is running the replaced PE on their system through SMB (direct execution without copying it down to their machine), and Pandemic is uninstalled, this can cause process instability for the replacement PE. The instability may not appear if the entire replacement PE is cached on the target user's machine. Instability may not happen immediately if it does occur.
- (S//NF) If the replacement PE file has a different icon than the target PE file, the remote users may not see the correct icon all the time. The local Windows machine tends to cache icons, and this icon caching can persist for some time (or until reboot). The best work around is to ensure that if the target PE file has icon information, the replacement PE file should then share that same icon information.
- (S//NF) There will be some memory leakage as a result of making 'safe' choices vs. potentially destabilizing choices. Typical memory leakage size will be around 1 KB per run of Pandemic (of NonPagedPool). According to various sources, Vista+ 64-bit systems cap NonPagedPool to 75% of RAM. So on a server with 64 GB of RAM, NonPagedPool is limited to 48 GB.
 - To limit leakage, run Pandemic with long death-timers. Avoid running Pandemic many times using short death-timers. If this becomes an issue (target server doesn't reboot often, operation requires many runs of Pandemic on a short leash) then the developer can investigate ways of reducing the memory impact

- (S//NF) Two different remote users that share the same machine (does not apply to VMs on the same machine), but log into the Pandemic machine using different user accounts (different SIDs) could cause targeting issues. If user account A on the remote machine is targeted, but user account B on the same machine is not, then the following issue can occur:
 - User A is running WinHex.exe, the targeted application, directly from the Pandemic File Server (PFS). User A really is running a Trojan'd copy of WinHex.exe. User B logs into the PFS and also directly executes WinHex.exe. User B, while not targeted, will still receive the Trojan'd WinHex.exe. This is because the machine that User A and User B share is caching the file.
 - The following scenarios will not//not trigger the issue:
 - User A is running WinHex.exe, the targeted application, directly from the Pandemic File Server (PFS). User A really is running a Trojan'd copy of WinHex.exe. User A is running off a VM on the remote machine. User B then gets on the remote machine and logs into the PFS using a separate VM, or the machine itself. User B will get the correct version of WinHex.exe
 - User A and User B are on different remote systems
 - User A is running the Trojan'd WinHex.exe. User B then gets on the same machine, and downloads WinHex.exe from the PFS to the local machine before executing the local copy. User B will get the correct copy of WinHex.exe.