ShimBad the Sailor

hexacorn.com/blog/2020/03/18/shimbad-the-sailor

March 18, 2020 *in Anti-Forensics, Autostart (Persistence), Code Injection, Living off the land, LOLBins*

Application Shims have been extensively covered by security researchers – a very comprehensive overview of the available techniques was presented at BH2015 (PDF warning) by Sean Pierce (@secure_sean who also happens to host a page dedicated to the subject at https://sdb.tools/).

I wondered if we could look at shims from a slightly different perspective, and this post is about it.

What if…

…we didn't change anything, didn't add any new entries, no custom databases etc.

What if…

We analyzed the existing shims and identified some that could do some interesting things for us? We would then need to fulfill the conditions required for shim to be triggered, and voila… we could now do things via a covert channel – that is, shim engine could be doing the dirty deed and a casual observer would be none the wiser.

Demo time.

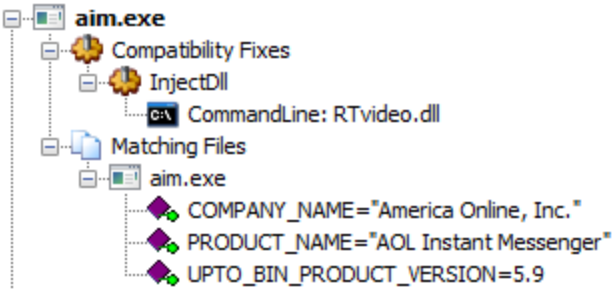
On Windows 7, AOL Instant Messenger can be loaded via *aim.exe* with following versioninfo properties:

- CompanyName = America Online, Inc.
- ProductName = AOL Instant Messenger

When system detects such program it applies a SHIM:

The shim loads a library *rtvideo.dll*.

I took a basic example from masm32 package and changed the properties of the file accordingly:



```
VALUE "CompanyName", "America Online, Inc."
VALUE "FileDescription", "Blank Dialog Template"
VALUE "FileVersion", "1.0"
VALUE "InternalName", "basicdlg"
VALUE "OriginalFilename", "basicdlg.exe"
VALUE "LegalCopyright", "© 1998-2011 The MASM32 SDK"
VALUE "ProductName", "AOL Instant Messenger"
VALUE "ProductVersion", "1.0"
```

and then compiled, renamed to aim.exe and the phantom DLL was added to the program by the shim engine.

Time ...	Process ...	PID	Operation	Path	Result
5:30:4...	aim.exe	164	CreateFile	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	QueryBasicInfor...	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	CloseFile	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	CreateFile	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	CreateFileMapp...	C:\test\rtvideo.dll	FILE LOCKED WI...
5:30:4...	aim.exe	164	CreateFileMapp...	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	Load Image	C:\test\rtvideo.dll	SUCCESS
5:30:4...	aim.exe	164	CloseFile	C:\test\rtvideo.dll	SUCCESS

This is just a basic example of what is possible/available.

Some of the shims create files, rename them, modify stack, fake reading files, etc. etc. . This offers a gamut of possibilities that are worth considering from various perspectives:

- anti-sandbox, anti-analysis tricks
- capture the flag tricks
- after building a repo of shim gadgets one could potentially deliver a lot of functionality by using dummy, non-malicious files ran in a proper sequence
    - copy files
    - patch bytes (<win10)
    - load DLLs
    - run executables

- the example with *aim.exe* is truly fascinating as it represents a possibly novelty type of code injection: phantom sideloading
    - we sideload that DLL with a predetermined name w/o calling any obvious function inside the .exe
    - in the example I am using a custom *aim.exe* that is just quick & dirty piece of test code; one could potentially find that legitimate, original *aim.exe* and play with that
    - the latter could be potentially signed
    - and even better, could be not even directly referring to *rtvideo.dll*
    - as such, it could be a signed .exe phantom sideloading a DLL with a predetermined name — and in some cases becoming a potential phantom lolbin as well
- persistence is there too to consider

Now, this might have sounded a bit rosy, but reality is that analysing shims is a bit of a pain & options they offer are still pretty limited. Yes, the number of really useful shims is pretty low, let alone these that could be meeting all the cool requirements I listed above… As such, defenders shouldn't worry about this trick too much… Until this topic is explored a bit more