How to hunt: The masquerade ball

By ByPaul Ewing

Published: 2016-10-31 · Archived: 2026-04-06 03:14:31 UTC

Editor's Note: Elastic joined forces with Endgame in October 2019, and has migrated some of the Endgame blog content to elastic.co. See Elastic Security to learn more about our integrated security solutions.

Masquerading was once conducted by the wealthiest elite at elaborate dances, allowing them to take on the guise of someone else and hide amidst the crowd. Today, we see digital masquerading used by the most sophisticated as well as less skilled adversaries to hide in the noise while conducting operations. We continue our series on hunting for specific adversary techniques and get into the Halloween spirit by demonstrating how to hunt for masquerading. So let's start the masquerade ball and hunt for a simple but more devious defense evasion technique.

Defense Evasion

In nature, camouflage is a time-proven, effective defensive technique which enables the hunted to evade the hunters. It shouldn't come as any surprise that attackers have adopted this strategy for defense evasion during cyber exploitation, hiding in plain sight by resembling common filenames and paths you would expect within a typical environment. By adopting common filenames and paths, attackers blend into and persist within environments, evading many defensive techniques.

Part of the attacker's tradecraft is to avoid detection. We can look to frameworks, like Mitre's ATT&CKTM, to guide us through the adversary lifecycle. We've shown how it's useful for hunting for persistence (as our COM hijacking post demonstrated) and it also covers the broad range of attacker techniques, including defense evasion. DLL search order hijacking, UAC bypassing, and time stomping are all effective for defense evasion, as is the one we will discuss today - masquerading.

Attackers use these defense evasion techniques to blend in, making them easy to miss when hunting, especially when dealing with huge amounts of data from thousands of hosts. Let's start with some DIY methods to hunt for masquerading, which require an inspection of persistent or running process file names or paths.

The Masquerading Approach

We previously explored hunting for uncommon filepaths, which is a simple approach for detecting suspicious files. We can expand on this method by understanding masquerading. Let's focus on two different masquerading techniques:

1. Filename masquerading where legitimate Windows filenames appear in a non-conventional location.
2. Filename mismatching where filenames on disk differ from those in the resource section of the compiled binary.

Filename Masquerading

For filename masquerading, you need to first build the list of files which have masquerade potential. We'll call that the anchor list. A good approach is installing a clean base image representative of your environment (a fresh install of Windows will do). Next, you need to choose which files you care about. Like most things, there is a lazy approach and an approach that takes a little more effort, but will probably give you more meaningful results with less noise. To build your anchor list the lazy way, simply enumerate all files in C:\Windows including the filename and path and use that as your anchor list.

However, there are a huge number of filenames in this list, and you should ask yourself questions about the likelihood of an adversarial masquerade before putting it in the anchor list. After all, it isn't much of a masquerade if the legitimate filename seen in a process list or anywhere else might cause someone to question its legitimacy, even if it's a system file, such as NetCfgNotifyObjectHost.exe. So, put in a bit more work and make a custom list of native Windows files, such as svchost, lsass, winnit, smss, and logonui, which show up constantly and are likely to be passed over if an experienced but rushed investigator is inspecting the name. It is also a good idea for the anchor list to include names for other common applications you expect to find in your environment, such as reader_sl.exe, winword.exe, and more.

Once the anchor list is compiled, you can start using it during your hunt operations. List the running processes, persistent files, or some other file-backed artifact you're interested in. Compare those names to the anchor list. Do the filenames match? There will be many matches. What about the filepaths? If not, you know where to target your hunt. There are legitimate reasons for this happening (users do unexpected things), but locating this simple defensive evasion technique is a good way to find intrusions.

We'd also recommend some additional triage of results before calling this a legitimate detection and embarking on an incident response. Easy things to do include checking hashes against the masquerade target in the anchor list. If it's a match, it's probably a false alarm, and check the signer information for the file as we discussed in the previous post. Be sure to avoid being too trustworthy of the name on the cert, as actors sometimes can get code signing certs that look similar to something legitimate...but that's a topic for another day.

If you find this approach worthwhile, you will have to keep your anchor list updated. Software changes and if you don't change with it, you'll have gaps in your analysis.

Filename Mismatch

Why stop at simply comparing files to your anchor list when more can be done? In this bonus masquerading approach, let's look at filenames on disk and from the resource section of the binary. There's a wealth of additional information here, including the MS Version info. As they note, it includes the original name of the file, but does not include a path. This can inform you whether the file has been renamed by a user.

Obviously, if the filename on disk doesn't match the original file name, there are generally two possibilities: either the user renamed it, or maybe someone brought a tool with them, but doesn't want you to know. Let's take DLL implants for example. Many APT groups have brought rundll32 with them, as opposed to using the native Windows version. APT groups aren't the only ones masquerading. Everyone does this!

Endgame @ the Masquerade Ball

Crafting your own anchor list, regularly updating it, and manually comparing the list to your hunt data or adding this analytic to your bag of post-processing scripts may work for some, but it calls for routine grooming. Let’s take a look at how easy it is to hunt for masquerading using Endgame, where we provide this as one of the many one-click automations in the platform.

Conclusion

Whom amongst us doesn’t love to use Halloween as an excuse to masquerade as someone, or something, else? Unfortunately, adversaries embrace this mentality year round, hiding in plain sight, actively evading detection, and trying to blend in. Clever use of masquerading within filenames can make their activities difficult to detect. While there are manual means to detect mismatches and masquerading, this can be time intensive and may not scale well to larger environments. Thanks to Endgame’s advanced detection capabilities, in a few clicks we are able to quickly catch those masqueraders, remediate the intrusion early, and get back to the ball.
