Does Application Whitelisting work?

Created by Dennis Underwood, last modified on Oct 23, 2023

What Is Application Whitelisting?

Application whitelisting is a concept in which you have a list of programs that are allowed to operate, and anything else is not allowed to function.

The most advanced application whitelist technologies will combine some type of source and destination logic. For example, an application whitelist may say, “Windows notepad is allowed to access the Taxes folder on my desktop, but Microsoft Word is not.”

We will assess this further, but generically, application whitelisting defensive strategies rely strongly on these two variables:

1. The attackers agree to follow the rules you set out for them with respect to your application whitelisting cybersecurity tool.
2. Your environment is very small with few applications that remain pretty stable, and without exploits.

Drawbacks to Application Whitelisting Technologies

Complexity

IT Networks are like complex organisms

Even small IT networks are constantly changing entities with a myriad of behaviors. Applications are constantly being added and removed by users. Modules inside of applications applications are constantly being added and removed by the program creators, especially on update. Lastly, application behaviors, in addition to changes in code resulting in behavior changes, are used in

different ways by different users at different times. Now imagine the team that would have to support tracking that constantly, and the IT help desk tickets that would have to come in constantly.

Realistically, the only stable IT environment is one driven by kiosks, in which programs are installed without the ability to upgrade, be added, or removed, and users are narrowly allowed to do only one or two functions. Maybe a kiosk for photo printing or something. In that case, application whitelisting would only need to be updated every time the kiosk terminal is upgraded twice a year or something.

More Advanced Application Whitelists Means More Complexity

The simplest application whitelists simply have lists of programs that are allowed to exist in a user's system.

This in itself can be challenging to manage, but leads to a variety of potential enhancements.

For example, just because the payroll department has an application to interact with the company's paycheck system, shouldn't there be a separate set of rules for the warehouse team, who has different (non-payroll) applications?

So, enhancements around which divisions or users can run which applications are an obvious enhancement. An enhancement that now has created additional complexity, in addition to just tracking the number, type, and versions of programs on your network.

Now, let's say that we need another set of enhancements - we need to decide which user has access to which application, for which set of data.

That, also, makes complete sense, right? You don't want to have your warehouse team have access to the HR records, and the HR team probably doesn't need access to the warehouse delivery team's operations data.

That is a ridiculously complex set of combinations to keep track of, now that data locations have been added as a variable.

So - with each necessary enhancement to make application whitelisting technologies effective, the management, configuration, and ongoing (arguably never-ending, always escalating) support needed to keep it functional makes the technology unusable. The result is normally that large groups of users are given access to large groups of applications, and likely almost all of the data. So - a nearly useless configuration of the technology, because no company has the resources to keep it managed properly.

Applications Interact With Each Other

It is common in modern operating systems for applications to interact with other applications. Sometimes, the interaction is due to an integration between two applications, that enhances the user experience - like a Word document opening up Microsoft Photos. At other times (this is very common) there are calls between programs that are done in a way that is typically invisible to the end user.

Revision #1
Created 2026-05-18 20:28:12 UTC by Dennis Underwood
Updated 2026-05-18 20:28:12 UTC by Dennis Underwood