

Application Whitelisting

- [Does Application Whitelisting work?](#)
- [What is Application Whitelisting?](#)
- [How Can I Tell What Would Have Been Suspended Except for the Tailored Behavior](#)
- [How to Manage Tailored Behaviors](#)

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 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.

What is Application Whitelisting?

Created by Dennis Underwood, last modified on Aug 22, 2022

- [Introduction](#)
- [Types of Application Whitelisting - a non-technical description](#)
 - [The Obvious & Cumbersome Hall Pass \(a good thing\)](#)
 - [The Generic Hall Pass \(probably not a good thing\)](#)
 - [The Generically Specific Hall Pass](#)
 - [The Specific-Specific Hall Pass](#)
 - [The Compromised Application](#)
- [In Summary - Questions to Ask When Investigating an Application Whitelist Capability](#)
- [Deep Reading from the National Institute of Standards & Technology \(NIST\)](#)

Introduction

Application whitelisting, in Cyber Crucible's context, is giving an application permission to perform some action that otherwise might be considered suspicious or malicious.

Application whitelisting technology is typically found in behavioral analysis defensive tradecraft, though some attackers use it as well (not covered here) in their operations.

Think of it like at school, students were not normally allowed to walk the hallways during class. However, if they were given a "hall pass" in some form by an authority figure, students were not returned to their classroom or delivered to the principal's office.

Types of Application Whitelisting - a non-technical description

If we continue with the "hall pass" analogy, there can be different techniques that school administration officials have learned, which helped formalize the behaviors some students (definitely not our CEO Dennis) may have tried to get away with. These all have direct correlation to security operations.

The Obvious & Cumbersome Hall Pass (a good thing)

21987331.jpg?width=278

Ever remember getting a hall pass, or a key to some room, that was incredibly large and obvious?

There were a couple good reasons for that:

1. So it would not get lost.
2. To lower the time and energy investment from an authority figure, in quick verification of the student's approved activities.

Good software won't "lose" a whitelist, but application whitelist software should enable fast, resource-efficient validation that the application has been inspected.

The Generic Hall Pass (probably not a good thing)

This is probably the least useful to application monitoring and schools alike, but requires the least management or expertise.

In this case, the student is allowed to run their errand in the hallway. The teacher may know, for example, but there is no validation from hall monitors as to what the student is doing. Also, if there is attempted validation, there are two levels of considerable resource investment:

1. the level of effort to approach the student, and interrogate them as to their behaviors and intent
2. the level of effort to ensure the student is being honest, to validate the student's explanation

With applications, this is most often seen when a program is added to a whitelist, and there is no further validation of behaviors or allowed activities. Unfortunately, the ability to gain context to an application's behaviors are typically not available for acquisition if not tracked externally by some other framework.

The Generically Specific Hall Pass

Not to be confused with, "go ahead Application, do whatever you want!"

Let's go back to the restroom pass, because it is so appropriate here. Given the pass' visibility, it is relatively easy to assess what the student should be doing. Egregious behaviors out of line with "going to the restroom" behaviors may be assessed without much difficulty.

Applications whitelisting which has generic boundaries of activities are the most common, though in many cases those boundaries are the equivalent of allowing a student with a restroom pass to play on the playground.

The Specific-Specific Hall Pass

Not to be confused by the "generically specific" hall pass!

22642689.jpg?width=340

While, with the generically-specific hall pass, we know the student is using the restroom - in a large school, we do not know

1. Who gave the student permission
2. Where they just came from.
3. Which restroom the authority figure had in mind when they gave approval.
4. How long the student has been "out and about".

This required more work on the authority framework for the school.

To return to application whitelisting, behavioral analysis is more accurate with more variables added to any decision-making capability.

Additionally, while investigating a hall pass that has been issued, you normally do not need to trace back a chain of 5 teachers to get to the source. You also can almost always trust the teacher wasn't some type of "bad teacher" that is telling students to behave improperly. Application whitelisting, in a specific-specific whitelist example, requires working backwards, to examine every parent or ancestor application that opened the next application. For example, a piece of malware may have opened up a legitimate Windows utility.

The following questions are roughly equivalent to the "student hall pass" above:

1. What was the application started for, or told to do? We need to know the intended behavior.
2. Who started the application? This is typically another program.
3. Which "parent" or "ancestor" programs resulted in this program opening? What were each of them doing, or intending to do?

The Compromised Application

Remember in various science fiction, or horror literature and movies, when an imposter has somehow taken

22413315.jpg?width=226

over control of a character's behavior? In schools, there isn't much chance of an instructor being taken over by aliens.

Application takeover is a favorite technique of attackers because, like in the movies, most observers see nothing amiss until it is too late.

In this case, an attacker takes a running program, that is currently loaded into memory. So, the fully intact program in the file system is ignored. The program in memory is edited to add the attacker's code. Much like an alien inside a human's body. The attacker does their criminal behaviors, and when the program is done running, most of the evidence is gone. When the legitimate program runs again, if the attacker does not edit the code again, it is just as the original legitimate coders intended. Maybe a better analogy, in that case, would be a werewolf?

An application whitelist, then, needs extra work to ensure that the program, and any ancestors or parent programs, are intact without hacker editing the programs while they are running.

In Summary - Questions to Ask When Investigating an Application Whitelist Capability

Make sure to understand the boundaries of a whitelist allows, when discussing application whitelisting with your technology team or vendor.

1. Does this capability examine and track what a program is supposed to do, and compare expected vs observed behaviors?
2. How much detail does the whitelist capability capture in its decision making? More, or less, than a detailed student hall pass?
3. Does the whitelist capability track and investigate parent and ancestor programs for maliciousness? (Program A opens B, opens C, opens D - very common)
4. Does the whitelist capability validate the running program has not been tampered with?

Deep Reading from the National Institute of Standards &

Technology (NIST)

An authoritative publication that is not for casual reading, but is very descriptive.

While it does not cover every scenario, and specifics are not updated frequently enough to keep up with every scenario, the descriptions are strategic enough to be valuable in most circumstances.

The Cyber Crucible team always values insight from NIST publications, and sets aside time accordingly.

<https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-167.pdf>

How Can I Tell What Would Have Been Suspended Except for the Tailored Behavior

Created by Mark Weideman, last modified on Apr 30, 2024

Users can see what extortion responses would have been suspended if a tailored behavior did not exist by first going to the Extortion Responses page found under the Operations tab in the sidebar.

The Excluded column shows if an extortion response has been excluded or not, and the default column filter shows non excluded, non silent responses.

ExcludedColumn.png

To show excluded responses, open the Excluded column filter and select only Excluded.

ExcludedFilter.png

After clicking Apply, the grid will show excluded extortion responses that would have been suspended if the tailored behavior did not exist.

To see which tailored behavior was applied to a response, first click the arrow in the Number of Incidents column for the specific row. On the inner grid that reveals after clicking the arrow, the Hiding Rule column will show the name of the tailored behavior that was applied to the response.

ExcludedResponses.png

How to Manage Tailored Behaviors

Created by Mark Weideman, last modified on Sep 09, 2024

- [How to Create a Tailored Behavior](#)
- [How to Create and Edit Temporary Tailored Behaviors](#)
- [How to Delete Tailored Behaviors](#)
- [How to Copy Tailored Behaviors to a Different Group](#)
- [Creating a Tailored Behavior from the Extortion Response Page](#)

The Tailored Behaviors page can be found under the Operations tab in the sidebar. Note that users must have the “View Tailored Behaviors” permission for a group to be able to see the group’s tailored behaviors on this page

How to Create a Tailored Behavior

After navigating to the Tailored Behaviors page, click the exclude process icon above this grid.

Clicking this icon will popup a modal to create the tailored behavior.

CreateTailoredBehaviorIcon.png
Create Exclusion.png

Users have the option to specify a parent program with the exception

94732291.png?width=504
Enter additional restrictions to apply

CreateTailoredBehaviorAdditionalRestrictions.png

Users now have the option to limit the exception to specific file access triggers, all file access triggers are included by default. Note that only agents on version 4.4.6.2 and up have this file access trigger capability.

User also have the ability to create temporary behaviors, see more in the next section

How to Create and Edit Temporary Tailored Behaviors

Users can create a temporary tailored behavior by following the normal process to create a behavior as explained above, and then turning the “Make This Tailored Behavior Temporary” toggle on. Users are then able to enter the hours until the temporary behavior should expire (up to a week max).

Temporary Exclusion.png

The Behavior’s expiration date can be seen in the Expiration Date column on the grid.

To edit a temporary behavior to extend the expiration date, make a temporary behavior permanent, or make a permanent behavior temporary, click the edit icon on the desired behavior in the Expiration Date column.

Edit Temp Behavior Icon.png

Clicking this icon will open a modal where users can edit the expiration date of the behavior as desired

Edit behavior modal.pngScreenshot from 2024-09-09 15-21-58.png

How to Delete Tailored Behaviors

To delete a behavior, or multiple behaviors at the same time, first select the behavior(s) to delete on the grid and click the trash icon above the grid.

Second, type “delete” and click the Delete button.

RemoveTailoredBehaviorIcon.pngRemoveTailoredBehaviorModal.png

How to Copy Tailored Behaviors to a Different Group

To copy a behavior (or multiple behavior at the same time) to another group, select the behavior(s) to copy on the grid and click the copy icon above the grid.

Clicking this icon will popup a modal where you can select the group the behavior(s) should be copied to.

This modal also has an option to delete the existing behaviors in the selected group (except auto generated behaviors), and only contain the copied behaviors in the selected group after submitting the request.

CopyTailoredBehaviorIcon.pngCopyTailoredBehaviorModal.png

After submitting the request, the copied behaviors will now appear for the selected group.

ResultGrid.png

Creating a Tailored Behavior from the Extortion Response Page

Users have the ability to create tailored behaviors from the Extortion Responses page.

First, click the arrow on the desired extortion response row in the Number of Incidents column to reveal the inner grid.

Second, click the exclude response icon in the Response Name column, which will popup a modal to create the tailored behavior.

ExcludeResponseIcon.png

On this modal, you may create behaviors using the suggested executable paths or submit the response for review if you are not seeing what you need from the suggested paths.

Exclude Modal.png

You may also click the Take me to Tailored Behaviors Page button, which will redirect you to the Tailored Behaviors page and automatically prefill the create behavior modal with the same group, path, and program arguments from the response.

AutofilledModal.png

Note that by default the Limit Exception to Program + Parent Program is toggled off. The parent program path and arguments from the extortion response will also be filled out automatically if you toggle this setting on.

PrefilledArgs.png

The Tailored Behavior Exclusion after clicking Create with the Limit Exception to Program + Parent Program setting toggled on:

ExlusionResult.png