Equilateral of Exclusion Risk Framework

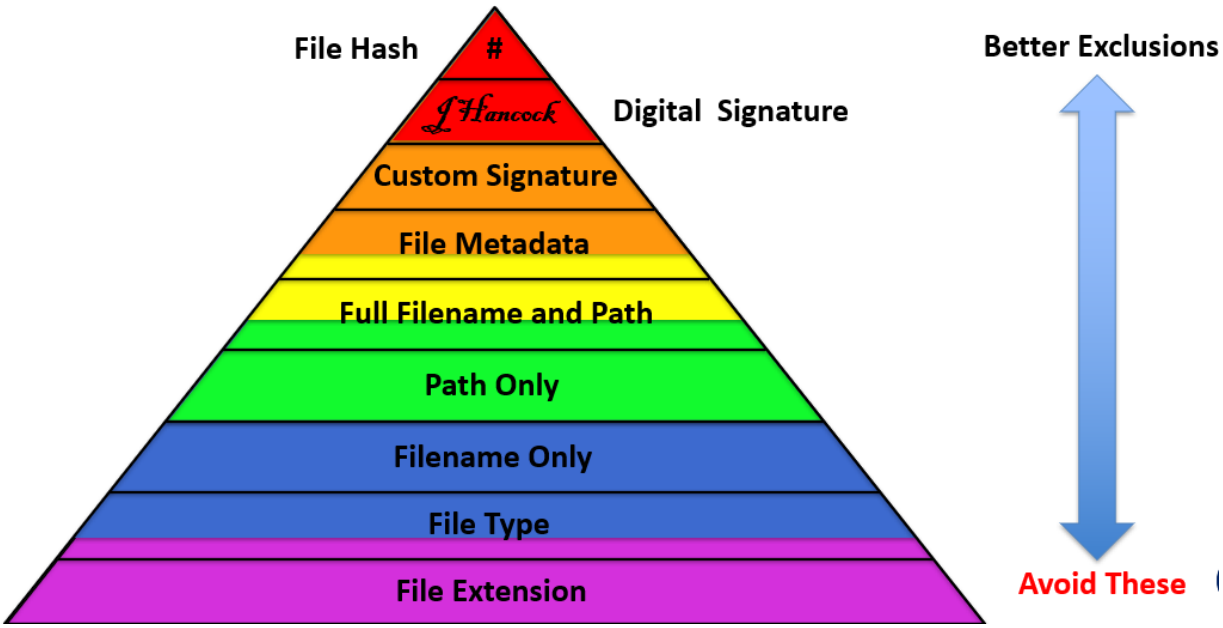
Jake Williams
@MalwareJake

Version: 1.0

Purpose

The purpose of this document is to explain the “Equilateral of Exclusion Risk” (EER) framework. Organizations frequently need to configure exclusions to avoid false positive alerts from security tools. While detection exclusions have always been necessary in security products, the need is increasing as endpoint security products collect more telemetry. If we concede that no detections are perfect and that some amount of data processed will have false positives, then we must conclude that more data will lead to more false positives. This, in turn, leads to a larger need for exclusions.

Endpoint security vendors offer a range of exclusions for their products. Those who must administer endpoint security products find themselves needing to configure exclusions that balance security with maintainability. Like most in security, these teams are asked to do more with less, creating an incentive to create whatever exclusions require the least maintenance. The EER framework seeks to provide, for the first time, an industry standard framework to understand relative risk of any given exclusion.



High-level Exclusion List

The following demonstrate the exclusions considered for version 1.0 of the EER framework:

- File Hash
- Digital Signature (in descending order of risk): certificate fingerprint, validated certificate subject name, trusted digital signature, digitally signed at all)
- Custom Signature Match/Yara Rule
- File Metadata
- Full Filename + Path
- Path Only (in descending order of risk): path only, path regex, path wildcard
- Filename only
- File type (validated by internal inspection of at least the file header)
- File extension

Future versions of the framework will address new exclusion options as they are added by endpoint security vendors.

Equilateral Order (in descending order of risk)

File hash

Benefits

Depending on the hashing algorithm used, there is a near-zero chance of a threat actor bypassing the exception. Chosen plaintext MD5 collisions can be relatively easily computed. SHA1 collisions have occurred, but require massive computational power and likely cannot use chosen plaintext. SHA256 collisions are computationally impossible at this time.

Drawbacks

File hashes are obviously extremely specific, so the initial work of creating exceptions for multiple executables for a given application will require significant effort relative to other exceptions. Every new version of the software needs a new exception, so maintenance is also relatively high effort. Additionally, every new version of the software will generate alarms or fail to execute until the exclusions are updated with the new file hashes.

Digital Signature

Benefits

Digital signatures can offer a mathematically verifiable proof of origin for a binary and as such are particularly useful for exclusions. Unfortunately, exclusion configuration options are not standardized across security products. When the chosen security product offers more than one exclusion type for digital signature, use the ordering in the notes section to understand the relative levels of residual risk.

While certificate fingerprints are most precise, using a validated subject name is much more manageable over time. The validated subject name likely only results in higher overall risk in the following situations:

- A threat actor adds a new trusted root to the compromised endpoint's certificate store (and that trusted root certificate is used by the security tool)
- A threat actor can use some means to obtain a certificate with the approved subject name (highly sophisticated and not in most threat models)

Drawbacks

The main drawbacks in using digital signatures for exclusions deal with how the security product itself handles verification of the signature. There are numerous examples of security products with logic errors in validating signatures and some where the product simply fails (by design) to validate the certificate at all.

Notes

Not all digital signatures exclusions are created equally. Make sure you understand the residual risk, depending on the type of digital signature exclusions supported by your system. Generally, in order of descending risk, the following apply:

- Certificate fingerprint
- Certificate subject name in a certificate with a validated trust chain
- Presence of a digital signature with a validated trust chain
- Presence of a digital signature with a trust chain that cannot be validated

Custom Signature Match

Benefits

A custom signature, such as those used by the tool Yara, offers a robust way for analysts to create very specific conditions for exclusion. A critical component for exclusions is the ability to configure exclusionary conditions in the definition, for instance "(A and B) and not C".

Drawbacks

The value of any exclusions using custom rules (and residual risk from the exclusion) is based on the quality of the rule. This, in turn, is usually driven by the experience of the analyst. The most important experience the analyst should have when configuring the logic of the rule is in offensive techniques. If the analyst lacks experience with offensive techniques, they should consult with Red Team assets to examine ways in which the rule can be bypassed and understand residual risk.

File Metadata

Benefits

If the security tool supports parsing file metadata for exclusions, file metadata can be used for exclusions. In general, some combination of file metadata, such as a combination of product name, publisher, and copyright is better than any single piece of metadata on its own.

Drawbacks

File metadata is trivial for threat actors to fake if they understand how exclusions are configured. When exclusions are configured using file metadata, avoid configuring exclusions that rely only on common publisher names that might be easily guessed by threat actors, such as “Microsoft.” If wildcards are supported for the exclusion, avoid the use of wildcards in the definition. Exclusions using file metadata are already particularly weak and adding wildcards further decreases its efficacy.

Not all metadata is created equally. Metadata that includes frequently changing data, such as product version numbers are difficult to maintain and offer less security than configuring exclusions via file hashes.

Full Filename and Path

Benefits

Exclusions based on filename and path vary in security, but are most effective for applications that are frequently executed. They are more secure than exclusions based on either just a filename or just a file path. A threat actor seeking to use this exclusion for a security product bypass must replace a legitimate executable, something likely to be noticed in any frequently executed application.

Drawbacks

Threat actors have been observed replacing legitimate executables and replacing them with malware. In more complex cases, threat actors have configured that malware to launch a subprocess with the original legitimate executable (presumably to minimize opportunities for detection). Before configuring this exclusion, inventory the behavior of the excluded binary. In particular, analysts should know whether the excluded program creates child processes and create threat hunting rules to audit for behavioral changes.

File Path

Benefits

File paths are common exclusions configured in security products. This is often performed for convenience, since a given application installation may run multiple executables from the same folder. If DLLs also need to be excluded based on security policy, file path exclusions are also extremely useful since many DLLs will typically be in a given program directory.

Drawbacks

File path exclusions are easily abused by threat actors. In particular, threat actors can trivially use file path exclusions to gain execution through DLL sideloading or path hijacking. Threat actors can also reasonably predict paths that may be excluded, increasing the risk of using file paths for exclusions. When recursive options are available path exclusions, they should be used with extreme care (if at all). Temporary and download directories should not be excluded using file paths. Regular expressions (when supported) are extremely granular but are also prone to misconfiguration (leading to additional residual risk).

Notes

When file path exclusions are used, note that wildcards, regex support, and recursive exclusions all increase risk. Generally, in order of descending risk, the following apply:

- Path only
- Path with regex
- Path with regex, recursive processing
- Path only, recursive processing
- Path with wildcard
- Path with wildcard, recursive processing

Filename Only

Benefits

Filenames are extremely fragile for creating exclusions and should only be used when other, more robust exclusion types are not available. Those considering this exclusion should only exclude relatively unique filenames, not something extremely common such as “svchost.exe” or “test.exe.”

Drawbacks

Filename exclusions are trivial for threat actors to abuse. Threat actors need only examine the process list on a system to enumerate possibly excluded filenames. If wild cards or regular expressions are supported for filename exclusions, they should only be used in extreme circumstances when no combination of other exclusion types will be effective.

File Type

Benefits

File type exclusions rely on internal file contents that specify its type, typically in a file header. This exclusion type should not be confused with file extension exclusions, which don’t inspect any internal file contents. This exclusion is most useful in situations where an application is regularly generating files with a custom file type containing data that is triggering false positive alerts.

Drawbacks

In no case should any executable file type or one containing interpreted code (such as macros) be excluded. When file type exclusions are supported by a security product, the definitions for what constitutes a file type are often not user-configurable. Because file type definitions are opaque to the end user, they are considered relatively fragile. However, if the end-user can configure file type definitions, file type exclusions can be made safer than path exclusions (particularly those using recursive options).

File Extension

Benefits

File extension exclusions rely only on the file extension portion of the filename in order to exclude the file from inspection. In most cases where this exclusion has been observed being

used in the production, a combination of other available exclusions would have been more appropriate.

Drawbacks

This exclusion is extremely fragile and is trivial for threat actors to abuse. There are very few cases where this exclusion should be considered. In no case should any executable file extension. File extensions typically containing interpreted code (such as macros) should also not be excluded. Before using this exclusion, get a second opinion from a senior security professional.

Conclusion

The EER framework defines risk levels associated with endpoint security exceptions. This should be used by security practitioners to understand levels of residual risk and to make appropriate choices when faced with multiple options for exclusions.