This feature is introduced in ISESteroids 2.0.10.24. To make sure you are using the appropriate version, when ISESteroids is loaded, enter:

PS> Get-PSXVersion

If this cmdlet is not found, then you either did not load ISESteroids yet, or you have an outdated version. Visit http://www.powertheshell.com/isesteroids2/download/ to download the latest version.

PowerShell Risk Assessment with ISESteroids 2.0

Sharing scripts and downloading sample code becomes common place. But how do you know a script is safe? You would have to read (and understand) it line by line, and carefully check that the script indeed does what it claims it does, before you run it.

The truth is that most people do not have the time (or the knowledge) to thoroughly check script code. Instead, they take even complex code, try and adjust it to their needs, then run it. This "prayer based script sharing" (PBSS) is risky, especially when you imagine that often these scripts are used in production environments.

At the European PowerShell Summit in Amsterdam, we had talks with Jeffrey and his team about how to help users quickly assess the potential risk arising from script code, and make this a safer place.

Today, I'd like to introduce to you a script risk assessment system built into ISESteroids 2 (introduced in version 2.0.10.24).

Quickly Assessing Risks

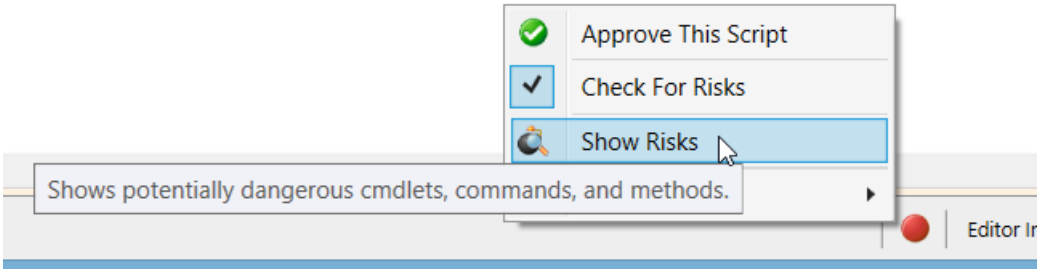
In everyday life, there must be a quick way for PowerShellers to know whether script code imposes risks and needs extended review.

Any script you open in ISESteroids will be internally analyzed, and the result appears in the status bar: you will see a green, yellow, or red icon.

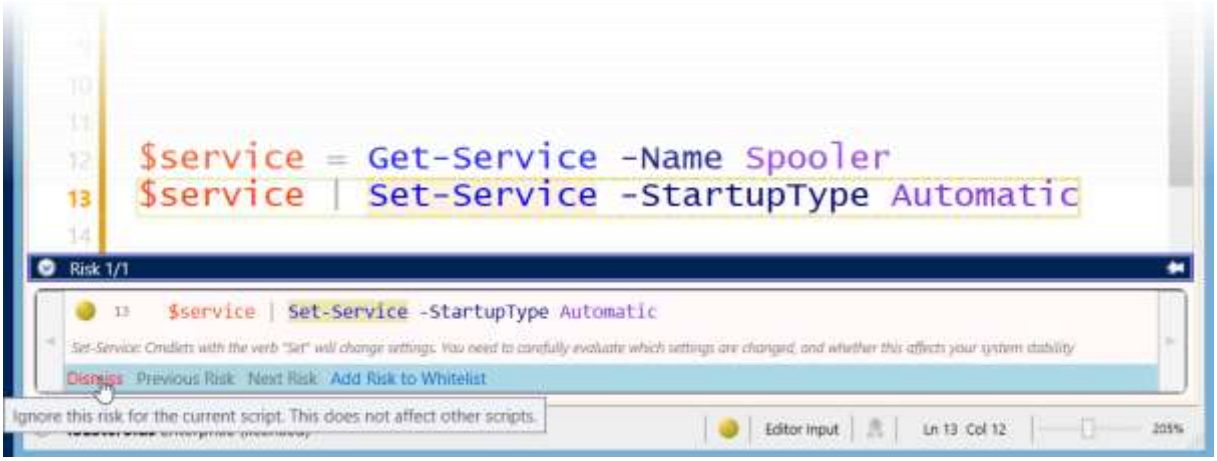
When you hover over the icon, a tooltip appears and tells you why ISESteroids thinks there is a risk, and what you can do about it. We'll dive into the risk evaluation engine in a second. At this point, the user is simply consuming risk assessment information and can quickly estimate whether a script needs review.

Reviewing Risks

To review and assess the potential risks in a script, the user can then click the icon to open a context menu with additional options. To start the review, you would choose "Show Risks".



This opens a sliding area beneath the editor and displays the first risk found. The editor will automatically move to the line containing the risk. The user can now quickly look at the actual code and determine whether or not this imposes a risk.



If the particular command is no risk, the risk can be dismissed by clicking "Dismiss". It will then be considered "safe" for this risk check only. The sliding area will automatically move on to the next risk (if any additional risks are left).

Or, the user can decide that this risk is really never a problem in his or her environment. So if the user wants to treat "Set-Service" as always safe, by clicking "Add Risk to Whitelist", the risk is placed on a white list and never be considered as risk anymore.

We will look at the white list and the additional ways to configure the risk engine just a bit later.

Approving Script

Once all risks have been evaluated, and no risk is left, the sliding area offers to approve the script. You can approve a script temporarily or permanently.

When you click “Approve”, the script is treated as “safe” on your machine only. The icon in the statusbar changes to a green checkmark. This information is persisted in an NTFS stream, so when you move the script to another place (or haven’t saved it on an NTFS file system in the first place), the approval information is lost once you close the ISE editor.

To permanently approve a script and give others a chance to trust your approval, click “Digitally Sign”. This will add a digital signature to the script file. If you do not own a code signing certificate, ISESteroids offers to create one for you.

While these self-signed certificates can be created by anyone (and thus may be considered “unsafe”), they are absolutely safe for the purpose of trusting code. If you sign your scripts with any certificate (whether self-signed or not), the certificate is unique and owned just by you. So others can identify that a particular script was approved by you. When they trust you, they can now trust your code as well.

Identifying Trusted Code

When you review and approve a script, then this approval is valid for you only. What is needed is a way to preserve the approval so users can start to trust each other.

Digital certificates have long been used to document trust. To trust, a digital certificate would have to be issued by a trusted root authority. Trusted root authorities are maintained by Administrators inside the Windows certificate management. This is too limited:

- It is complex to get trusted certificates. You may not own one, and commercial code signing certificates are expensive
- Certificate trust can be managed primarily by Administrators, not necessarily on a per-user basis.
- You cannot use simple self-signed certificates because they are not trusted by others.

This is why ISESteroids has enhanced this limited system. It offers you to create free self-signed certificates, and it can sign your scripts with self-signed or any other available certificate, placing a permanent digital signature in your script.

It also identifies digital signatures present in a script that you load. The “signature” icon in the status bar immediately indicates whether the certificate is trusted, and the script has not been tampered with. When you click the icon, a menu opens and allows you to view the signer so you can see who actually signed the script.

In addition, you now can manually choose to trust a certificate. This trust is completely independent of the Windows system. It merely is a list of trusted certificates, so you can choose whom you trust. To trust a certificate, click the signature icon in the status bar, and choose “Trust Certificate”. This will place the certificate on your trusted list.

This way, even with cheap self-signed certificates, you can choose to trust people, and whenever they post scripts to a repository, these scripts will display a green checkmark trust icon in the status bar.

Note that if the script content does not match the signature anymore, i.e. the script was tampered with, ISESteroids will not show the trust icon but instead display the actual risk level. So if someone tampered with originally signed scripts, you will notice.

Certificate-Based Auto-Trust

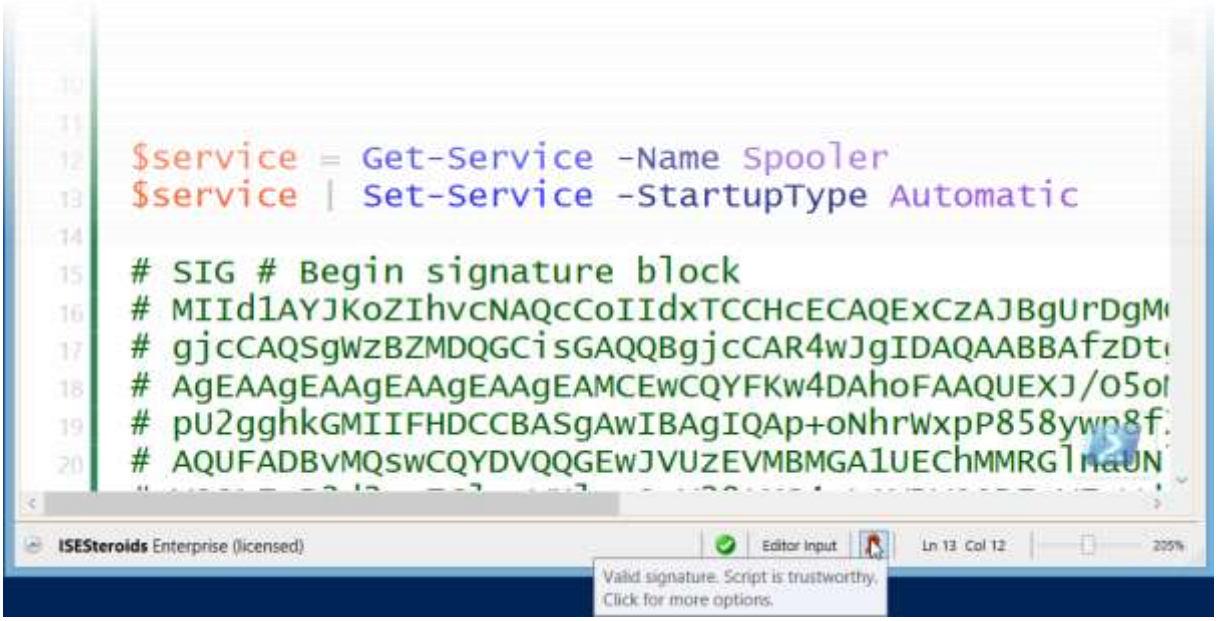
Whenever you load a script that has a digital signature, ISESteroids displays a “Code Signing” icon in the status bar.

The code signing icon quickly tells you whether a script has a signature, whether the signature is valid, and whether the script has been changed meanwhile.

If the certificate is trusted – either implicitly because of its root certificate, or based on your manual trust – the script will automatically be approved, and you see a green checkmark icon in the status bar.

You can always click this icon to open a menu and manually unapprove, check for risks, or choose “Settings” to disable automatic trust altogether.

Note that there will be no code signing icon until you save a script. “Untitled” documents are really no scripts and thus cannot be checked for a valid signature, even if they contained a signature block.



If the script was signed with a self-signed or otherwise untrusted certificate, the certificate still guarantees that the script was signed by a particular person, and that the code is still in the condition it was when the signature was applied. It is now just a matter of trusting the certificate.

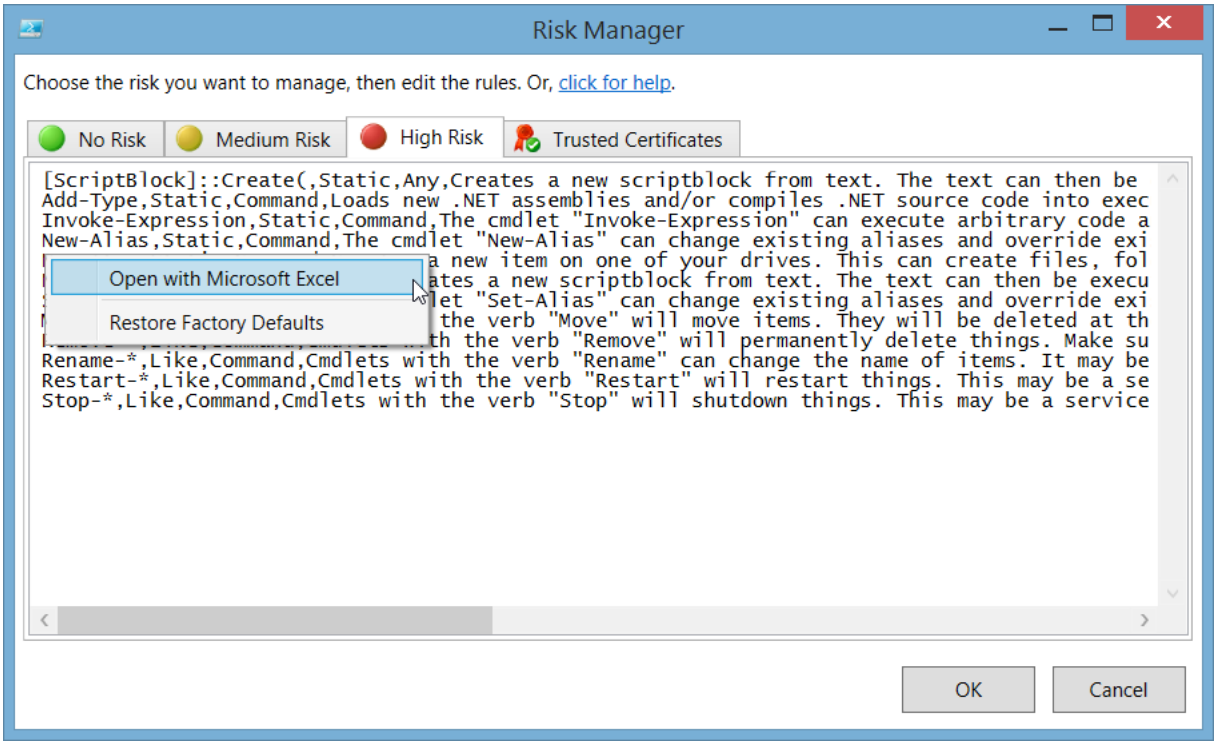
So if you know that a particular script comes from someone you trust, you can choose to trust its certificate.

You can add the thumbprint of that certificate to your trusted certificate list. Simply click the signing icon in the status bar to open a context menu, then choose “Trust this Certificate”.

Once you do that, any script signed with the certificate you trust will now be treated as “safe”.

Managing Risks and Trusts

The risk assessment engine in ISESteroids is completely configurable to your needs. To configure it, click the risk icon in the status bar, and choose “Settings/Manage Black/White Lists”. This opens up a dialog that lets you manage what is considered a risk. You can also manage the trusted certificates.



There are four lists for you to control risk assessment:

- No Risk: Anything in this list is considered “safe” and excluded from risk checks.
- Medium Risk: Anything in this list is considered a “medium risk” and displays a yellow icon
- High Risk: Anything in this list is considered a “high risk” and displays a red icon. Risks in this list are evaluated before risks in the list “medium risk”.
- Trusted Certificates: Contains the thumbprints of certificates that you chose to trust.

The easiest way to manage the lists is to right-click a list and open it in Microsoft Excel. If you do not own Microsoft Excel, you can edit the lists directly, too.

Note: When you edit a list in Microsoft Excel, make sure you close the edited file. Microsoft Excel locks files, so while a particular list is open in Excel, ISESteroids cannot edit it, for example add new entries to the white list.

How Risks are Defined

Each entry in one of the risk lists has up to four columns.

- SearchTerm: this is the text the risk manager is looking for. You can provide literal text, use the wildcard “*”, or provide a regular expression

- SearchType: this declares how the search term is treated. Allowed values are “Static” (use exactly), “Like” (allow wildcard “*”), “RegEx” (treat as regular expression)
- Scope: this declares where the search term is searched. Allowed values are “Command”, “Method”, and “Any”. “Any” will search the search term anywhere in a command expression.
- Description: Optionally, add a textual risk description that appears in dialogs and tooltips, and helps users understand what the particular risk is

When you edit any of these lists, and then click “OK”, the lists are saved. ISESteroids automatically sorts the lists. Lists are sorted by search type first, then by search term. Risks declared as search type “Static” are evaluated before any other because they are most specific.