# PowerShell Security Settings and Configurations

## Table of Contents

- PowerShell Security Settings ........................................................................................................ 2
- Execution Policy .......................................................................................................................... 3
- Additional PowerShell Security Settings ......................................................................................... 5
- Some Scripts in PowerShell ............................................................................................................. 7
- 10 Popular Scripts ........................................................................................................................... 8
- Script Resources for Security .......................................................................................................... 9
- Summary ....................................................................................................................................... 10
- Notices ........................................................................................................................................ 12
PowerShell Security Settings

PowerShell “execution policy”

- Determines how (or if) PowerShell scripts run
- Default setting is **Restricted**
  - Meaning scripts will not run; even locally created scripts.

```
PS C:\MyScripts> Get-ExecutionPolicy
Restricted
PS C:\MyScripts>
```

**064 If I want to know what my execution policy is, the command is Get-ExecutionPolicy. And you can see that here my execution policy is restricted. And that simply says, as I mentioned, that's the default. You're not going to run scripts if they're not going to be signed. Even scripts that you create locally, if you create it yourself, and you don't sign it, restricted says they're not going to run. So, you would have to figure out how to sign your scripts. All right?**
**Execution Policy**

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restricted</td>
<td>Does not load configuration or run scripts <strong>Default</strong></td>
</tr>
<tr>
<td>AllSigned</td>
<td>All scripts and configuration files must be signed by a trusted publisher.</td>
</tr>
<tr>
<td>RemoteSigned</td>
<td>All scripts and configuration files downloaded from the Internet must be signed by a trusted publisher.</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>Loads all configuration files and scripts. Attempting to run downloaded, unsigned scripts results in a permission request.</td>
</tr>
<tr>
<td>Bypass</td>
<td>Nothing is blocked; no warnings or prompts.</td>
</tr>
<tr>
<td>Undefined</td>
<td>Removes current execution policy from the current scope <strong>default scope is the local machine</strong></td>
</tr>
</tbody>
</table>

**065** Now, you could change your execution policy. And here we give you an example of what some of the choices are. Let’s see. Remote signed, all scripts and configuration files downloaded from the Internet must be signed. That kind of implies that I can run my locally created scripts even without signing them. So, if you want to run just locally created scripts without signing them, remote signed is okay.

All signed, all scripts and configurations must be signed by a trusted publisher. And so, probably unrestricted is the least desirable for
us because that allows us to run pretty much any scripts that we want, effectively.

Yes?

Student: I seems to me if you had a script that was useful, but couldn't run it, and it was short, couldn't you just run the command itself and bypass the signing, just running the command in PowerShell itself?

Mark Williams: Absolutely, yes. Yes, you can.

Student: And then just skip the stepping through this command.

Mark Williams: Right. This is just for the scripts themselves. Yeah, there are a number of scripts that are just a handful of commands. As a matter of fact, a lot of the things that we just did as far as adding account and stuff, there are scripts for doing that.

For example, adding Barney Rubble into the engineering group, that's easy to do for one person. But if I had ten, fifteen people to add in, I would probably want to script it, script that and have that script point to a list of here are my ten or fifteen people.

And if I wanted to add those people to-- if I wanted to create multiple accounts, doing it once or twice, not a big deal. But then if I want to do a hundred, now a script comes into play. And so, each of those
commands, short command by themselves, I could just do them individually, but scripting certainly makes it easier.

Additional PowerShell Security Settings

Script extension is .ps1

- .ps1 is not registered as an executable within Windows.
  - So, by default double-clicking a .ps1 file, the script does NOT run.
- Does not search the current path for scripts
  - To execute scripts in the current folder, the user must type 
    ./[scriptname]
  - Helps prevent command hijacking

**066 Scripts in PowerShell have an extension of .ps1. It is not a registered executable within Windows. So, if you were to, through Windows explorer, find a script and double click on it, nothing's going to happen. That script is not going to run. That's really nice for us. It's a safety mechanism.

Another safety mechanism is we're not going to, by default, search our
local directory, the current path for a script because sometimes what people might end up doing is taking a common command like dir, dir is-- and they might make a script called dir.ps1. And so, that ps1, who knows what it might do. It might dump all of your users. Or it might add you to a different domain. Who knows what kind of malicious stuff somebody might do with a script? So, if I were to type in dir, and there was a dir.ps1 in my path somewhere, and it would hit that, then that particular script would run. So, they say we’re not going to search the path. We’re not going to execute and .ps1s that are in our path. So, if you want to execute at .ps1, you’re going to have to type in the dot slash to say execute that script from my current folder where I’m at because we just don’t search the path at all, which means your current folder’s not looked at. It’s going to say I don’t know where it is, what are you talking about. Does that kind of make sense? It helps prevent command hijacking. All right?
**067** Here are some sites or locations that Microsoft—actually, these aren't locations. These are specific scripts that are helpful. If you want to add a bunch of users to a group, Microsoft provides a signed script for that. If you want to list, here are some of the inactive accounts in active directory, another signed script there. Installing SharePoint on Windows 7, Microsoft provides a script. All this stuff is available through the repository at the scripting guy's location on Microsoft.com. So, we just gave you an example of some of the scripts that you might be interested in.
10 Popular Scripts

**068 From the scripting guy's blog, he has a list of the ten most popular scripts that are out there. I guess, if you want to, we can go take a look at his blog and see what other information is available. But ten most popular scripts that people are interested in using are there.
**069 And then, in this particular case, this is a list of scripts that are specifically designed for the security professional in mind. I don't know what it is that you, as a security professional, might want to do. But they have seventy-seven different scripts specifically for security related tasks and functionality at the scripting guy's website. So, just about anything you want to do is probably somebody's already figured out how to do it for you.
**Summary**

PowerShell Description

Accessing PowerShell

Basic Scripts in PowerShell

Security Settings and Configurations

---

**070 So, PowerShell, it is a very robust tool to accomplish just about anything you want to do in the command line. Administer users and permissions, administer the network, administer software, ACLs, you name it, we have the ability to do it in PowerShell. So, I don't know. Do we have any questions about what we covered in PowerShell?**

Student: I think it's really good. It does a lot of stuff we--

Mark Williams: It's a super powerful tool.
Student: All the stuff we did in UNIX because I know they really didn't want to have the command-- they didn't want any command line in it originally. But everyone kept using it. And they just couldn't get rid of it. So, now they gave in.

Mark Williams: Well, if you're talking about a single computer, nobody wants the command line for a single computer. But if you're talking about an enterprise environment, command line is almost an absolute must for an enterprise environment because you cannot script pushing a button. And the GUI trying to do anything-- doing a lot of the things through the GUI, it takes up a lot of bandwidth to do it through the GUI. I mean running a script takes up very little resources, running a command line interface, very little resources. So, I think you're absolutely right. They gave in. and they gave us a very wonderful tool. I think PowerShell is super robust and super capable. And it'll be-- there's going to be a learning curve to it.

It's going to take a while before, especially some of the administrators that are hardcore using WMI, before they start moving over and-- or some of the other tools that are out there, before they really start to make PowerShell their tool of choice. But, yeah, it's very good.