

Securing Windows PowerShell

Jeffery Hicks
SAPIEN Technologies

Pre-requisites for this presentation:

1) Some PowerShell experience

Level: Beginner

Why Security?

- Why not?

PowerShell: Vulnerability?

- PowerShell is *powerful*
- Ad-hoc usage isn't that dangerous – you need knowledge & permissions
- Scripting is dangerous – social attacks to get users to run scripts are well-known
- So PowerShell's security focuses on keeping malicious scripts from running without significant effort on your part

Secure by Default

- By default:
 - PowerShell won't run scripts
 - When configured to run scripts, the shell can require that they be digitally signed
 - .PS1 filename extension not associated with the shell
 - Must specify a path in order to run a script

SBD: Won't run scripts

- PowerShell has an *ExecutionPolicy* which is set to Restricted by default – no scripts will run
- Other ExecutionPolicy settings:
 - AllSigned: Only signed scripts run
 - RemoteSigned: Local scripts don't need to be signed, remote ones (downloaded) do
 - Unrestricted: All scripts will run (bad idea in a production environment)

Modify ExecutionPolicy

- Get-ExecutionPolicy
- Set-ExecutionPolicy
- Or, better, use the ADM template for GPOs and set this centrally
- GPO overrides local settings
- Download ADM from Microsoft's Downloads site (search for "PowerShell ADM")

What is script signing?

- Signed .PS1 files contain a "signature block" in comments at the end of the file
- Intact signature means...
 - The script has not been modified since signed
 - The identity of the script author/signer is verified

Script Signing



Signing Details

- Class III Certificate
("AuthentiCode Code-Signing Cert")
- From a commercial CA or your own internal CA

Controlling ExecutionPolicy

- **Get-ExecutionPolicy** and **Set-ExecutionPolicy** cmdlets
- Download ExecutionPolicy ADM template (for Group Policy) from Microsoft (URL on class CD)

SBD: .PS1 association

- .PS1 filename extension associated (by default) with Notepad, not the shell
- Double-clicking a script file won't execute it, it'll just open it for editing
- Helps prevent scripts sent via e-mail from being accidentally executed a la VBScript

SBD: Must specify a path

- Create a script named `dir.ps1`
- Try to execute it by running **Dir**
- You can't: In order to run a script, you must specify a path - `./dir` will work
- Helps visually distinguish a script from a built-in command

Alternate Credentials

- Some PowerShell cmdlets have a **-credential** parameter which accepts either a username or a PSCredential object that specifies alternate credentials to use
- **Get-WMIObject** is a good example
- Providing a username launches a graphical dialog where you enter the password

How can I store the password?



- Write it on a sticky note and post it on your wall
- Seriously, just as safe as hardcoding the password

Storing Passwords

- Bad idea.
- NO safe way to do it. PowerShell knows this.
- So nothing in PowerShell accepts a stored password, just to make sure you don't try and do it

This Sucks.

- So wait, I have to type the password every time?
- Yes.
- Even if I'm connecting to a hundred computers.
- Well... okay, you can't store the password but you CAN store the CREDENTIAL

Storing alternate credentials

- Get-Credential prompts for a credential and securely stores it in a variable
- The variable can then be provided as the value of a `-credential` parameter
- Lets you create a credential once, and then use it multiple times
- Hint: Put this in your PowerShell profile to have it create the credential each time the shell runs



- A .PS1 file that is automatically executed each time the shell loads
- No profile exists by default
- Create
...\\Documents\\WindowsPowerShell folder,
create Microsoft.PowerShell_Profile.PS1
file
- Simply fill the file with commands that
you want run each time the shell loads

Profiles: The Security Gotcha

- Profiles don't exist by default
- Per-user profiles are (of course) writable by the user
- Malicious software can easily modify profiles
- Profiles run automatically when the shell starts
- See the problem?

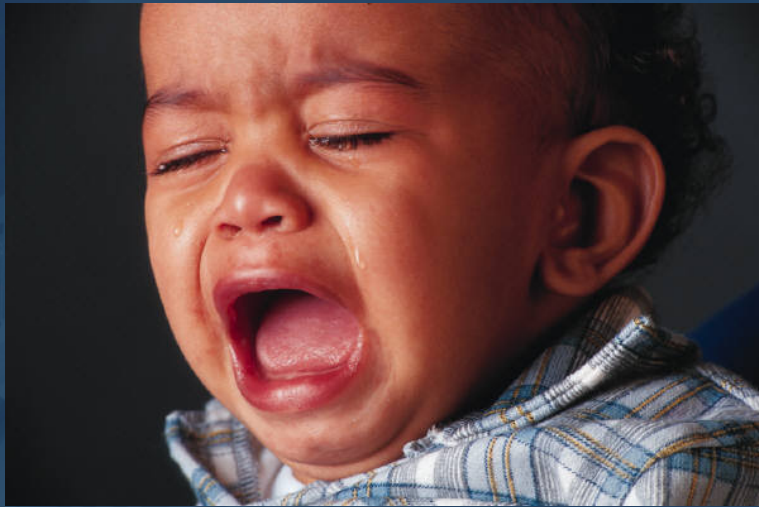
NOT OPTIONAL

- Profiles run automatically and present too great a risk
- AllSigned is the ONLY safe ExecutionPolicy in a production environment
- You don't let your kids play with handguns, you don't run less than AllSigned

Protect Just the Profile

- ACL the all-users profile so that only a never-used account can edit it
- In it, use Get-AuthenticodeSignature to check all per-user profiles for a signature
- Ensures that profiles are signed without requiring AllSigned to be used

But Signing is Too Hard!



- Time to be a grown-up IT boy or girl
- Win200x comes with CA software – you can implement it just for PowerShell signing

1. Install CA software (self-signed root)
2. Push out root cert through GPO (all machines trust)
3. Issue & install code-signing cert
4. Sign all scripts (PrimalScript does automatically or use Set-AuthenticodeSignature)

BTW... SecureStrings

- You CAN export a SecureString to an encrypted string and potentially store it in a file
- Reading it required the encryption key to be in clear text – so, zero protection
- Although you can only use the encryption key on the computer where it was created

- Secure by default!
- Vulnerabilities only come in when you alter the defaults...
- ...so it's your responsibility
- Securely store and re-use alternate credentials for cmdlets which support them

- blog.SAPIEN.com
- www.ScriptingAnswers.com
- www.PowerShellCommunity.org
- www.concentratedtechnology.com

- *Windows PowerShell v1.0: TFM 2nd edition* (Don Jones & Jeff Hicks)
- *Windows PowerShell Cookbook* (Lee Holmes)
- *Windows PowerShell in Action* (Bruce Payette)

Thank you

- jhicks@sapien.com
- Follow me: twitter.com/JeffHicks

