

Security Footprint

By Lee Holmes  
Winter Scripting Games 2014

Dr Scripto has an offer you can’t refuse - As the one-and-only Chief Resident Administrator and Scripter at your company, he’s realized that this ALSO makes you the Chief Resident Security Investigator. While he starts preparing a budget request for an active security defense infrastructure, he knows that things are in a pretty sad state of affairs. If somebody called today saying that their computer has started acting strangely, you’d be at an investigative loss trying to figure out exactly what’s changed.

To improve your organization’s ability to understand computer security-related investigations, Dr Scripto wants you to write a machine footprinting script.

This script should:

* Run as a daily scheduled task
* Extract and log important security data from the machine
* Upload the results to a centralized file share

There are lots of great online resources to help understand what is useful in a security incident response (“Threat Intelligence” and “Indicators of Compromise” are two common terms), but a good starting point is:

* Folder size and number of files
* Files: Path, Size, Last Modified, optionally file hash for specified folders
* Shares on the computer
* Installed software
* Running processes
* Running services
* Environmental variables
* Registry entries (especially those associated with “Auto run” functionality, such as HKLM:\software\Microsoft\Windows\CurrentVersion\Run) and the software settings in HKLM:\software

Your solution should be modular so that different parts can be run on different schedules if required

As you implement this footprinting script, here are some design ideas you should keep in mind:

* PowerShell excels at manipulating object-based data. If your script simply generates a huge text file, you’ll miss that opportunity.
* Your script should strive for efficiency and try to minimize both system impact and storage requirements.
* Your script should be easy to update. As you learn more about security investigations, it should be easy to add a new information source to your data collection capabilities.
* In many cases, the data generated by your script will contain sensitive information. Your solution should account for that.

Assume that you have permissions to the remote machines.

Your code should be production ready with:

* Ability to optionally report on progress
* Full error checking, reporting and handling
* Ability to accept pipeline input where appropriate
* Help is available
* Input is validated

In your entry submission, include a transcript that shows you running the command as described in this scenario.





## Key Criteria

These are some of the main items our judges will consider. You do not need to meet all key criteria, but you may earn extra points for doing so. This list is intended as a summary, and does not override the specifications of the scenario above.

* Consider the practices in *The Community Book of PowerShell Practices* (linked at http://powershell.org/wp/newsletter)
* Avoid aliases, except for –Object cmdlets; avoid positional parameters and truncated parameter names.
* Use appropriate error handling.
* Use appropriate means of displaying output, progress messages, errors, etc.
* When appropriate, manage pipeline input correctly
* When appropriate, validate input via parameter validation attributes
* Provide help for all scripts and functions, including examples
* Script filenames should include production date for versioning
* Use modular programming practices to maximize opportunities to share code
* Objects are output
* Processes included in output
* Services included in output
* Shares included in output
* Files and folders included in output
* Registry settings included in output
* Environmental variables include in output
* Installed software included in output
* Output is protected (encrypted)
* Code is modular
* Can set different schedules for different investigations
* Code is portable, maintainable (by others) and expandable

As is often the case in Windows PowerShell, there will be many ways to complete these objectives. In most cases, judges will prefer approaches that:

* Perform well under the load specified
* Leverage built-in functionality of Windows PowerShell rather than reinventing the wheel
* Are the most straightforward and easy to read and understand

