Scriptspråk

Kurs_NÄTD24LIN_SCSP25

Workshop 3: PowerShell - Active Directory Audit Report

https://github.com/muhad308/PowerShell---Active-Directory-Audit-Part-3

1. Problem Solving and Challenges :-

What were the biggest challenges you encountered in this workshop part?

How did you solve them?

- The biggest challenge was handling inconsistent or potentially invalid data from the JSON file. For example: If a `lastLogon` or `passwordLastSet` date was in an unexpected format the entire script would fail.
- I implemented error handling with `try-catch` blocks within the loops and filters.
- **Example:** In the `Get-InactiveAccounts` function and when filtering for `$oldPasswords`, I wrapped the date conversion `[datetime]$_.lastLogon` in a `try-catch` block. If it fails for a specific user, it logs a warning and skips that user instead of stopping the entire script.
- Another challenge was grouping and summarizing data in a readable report. Using here-strings (@"... "@) made it easier to format a multi-section report with tables and summaries.
  Finally, I had to account for the possibility that the JSON file might not include a computers array. I solved that with:

  ```
  if ($data.PSObject.Properties.Name -contains 'computers') { ... } else { ... }
  ```
- This check made the script flexible and reusable across different exports.

2. Learning and Development

What have you learned that you couldn't do before?

- I learned how to work with PowerShell objects more effectively using filters, calculated properties and structured reporting. Before this I didn't fully understand how JSON converts into PowerShell objects.
- Gained confidence using calculated properties in Select-Object (e.g. calculating password age).

Which concept(s) was/are the most difficult to understand and why?

- The hardest part was understanding how PowerShell pipelines and object-based data interact especially when mixing loops and Where-Object.
- Keeping track of object types while building the report.
- Calculated Properties (e.g., @{Name='PasswordAgeDays'; Expression={...}}) were initially tricky. Understanding how to create a new custom property on-the-fly within a Select-Object command requires shifting how you think about data pipelines.

3. Professional Relevance

How can you use these skills in your future role as a network engineer?

- As a Network Engineer these skills are crucial for infrastructure auditing, security and compliance.
- Scripts like this can generate daily reports on inactive users, old passwords or unused computers saving time and improving security. They are especially helpful during audits or when managing large AD environments.

Examples of real-world situations where this type of automation would be valuable:

- In a real-world setting this script could run as a scheduled task to alert admins automatically.

4. Code Quality and Improvements

If you were to do the workshop part again, what would you do differently?

- I would add CSV exports and a config file for the day thresholds. I would also include logging for better tracking.
- A configuration file for settings like the number of days (30, 90, etc.) instead of hardcoding them.
- Logging with timestamps using Start-Transcript for better traceability.

Which parts of your solution are you most satisfied with and why?

- I'm most satisfied with the **executive report section**. It combines summary statistics with detailed tables in a clear format. The use of PowerShell's formatting commands like Format-Table and Out-String makes the output look polished and easy to read.