Analysis Report for: tmp5c9fv_y2.ps1

This code is not C code; it's PowerShell. A C code analysis is not applicable. The provided script is designed to audit Active Directory for inactive users and computers. Let's analyze it as PowerShell code.

Overall Functionality

This PowerShell script queries an Active Directory domain to identify and export lists of:

1. **Inactive Users:** Users who haven't logged in within the last 6 months, excluding service accounts.
2. **Inactive Computers:** Computers that haven't logged in within the last 3 months, excluding specific patterns in their names (like "MASTER*" or "SRV*").
3. **Disabled Users:** Users who have been disabled for more than 3 months.

The results for each category are written to separate CSV files in the system's temporary directory.

Function Summaries

The script doesn't define any custom functions. It relies on built-in Active Directory cmdlets:

* **`Get-ADUser`**: Retrieves user objects from Active Directory. Parameters include a filter for selecting specific users based on properties like `Enabled` and `LastLogonDate`. It returns a collection of `ADUser` objects.

* **`Get-ADComputer`**: Retrieves computer objects from Active Directory. Parameters include a filter. It returns a collection of `ADComputer` objects.

* **`Where-Object`**: Filters a collection of objects based on a specified condition.

* **`Select-Object`**: Selects specific properties from objects in a collection, allowing for calculated properties.

* **`Export-Csv`**: Exports a collection of objects to a CSV file.

Control Flow

The script's control flow is primarily sequential. Each section performs the following steps:

1. **Define Thresholds:** Calculates dates representing the inactivity thresholds (6 months for users, 3 months for computers and disabled users).

2. **Retrieve Objects:** Uses `Get-ADUser` or `Get-ADComputer` to retrieve relevant objects from Active Directory, applying filters to narrow down the results.

3. **Filter Objects:** Applies `Where-Object` to further refine the results based on additional criteria (e.g., `LastLogonDate`, `lastLogonTimestamp`, `Enabled`, name patterns).

4. **Select Properties:** Uses `Select-Object` to choose the properties to be included in the output CSV. Calculated properties are often used to format dates or create new fields.

5. **Export to CSV:** Exports the filtered and selected objects to a CSV file using `Export-Csv`.

Data Structures

The primary data structures are the collections of Active Directory objects returned by `Get-ADUser` and `Get-ADComputer`. These are essentially arrays of custom objects, each representing a user or computer with its associated properties (e.g., `Name`, `SamAccountName`, `LastLogonDate`, `lastLogonTimestamp`, `Enabled`, `whenChanged`). The script also uses strings to store file paths and regular expressions.

Malware Family Suggestion

This script itself is not malicious. However, a malicious actor could adapt this script for malicious purposes. For example:

* **Reconnaissance:** The script gathers information about users and computers in the domain. This information could be used in further attacks.
* **Data Exfiltration:** Instead of exporting to a local temporary file, the script could be modified to send the collected data to a remote server, potentially exfiltrating sensitive information.
* **Credential Harvesting:** While this script doesn't directly attempt to harvest credentials, it could be extended to do so by incorporating other commands or techniques.

In summary, the script is a legitimate auditing tool, but its functionality could be abused for malicious purposes. It doesn't directly belong to a specific

malware family, but it demonstrates the kind of information gathering capabilities that malware often utilizes.