

Discovered Account Report Implementation Guide

26 Nov 2021

Ver 1.2

Table of Contents

Overview.....	3
About the Script.....	3
Support.....	3
Pre-Requisites.....	3
Environment.....	3
Knowledge.....	3
Required Access	3
Installation.....	3
Operational Overview	3
Implementation.....	4
Preparation.....	4
Operation	4
Miscellaneous.....	5
Data Output.....	5
Default Values	5
Known Issues	6

Overview

About the Script

This script is designed to get all accounts that were discovered from an Accounts Discovery scan, and format this account information into a readable CSV file.

Support

This script is not supported by CyberArk. By using the script, you agree to the disclaimer (shown when the script is executed).

Pre-Requisites

Before preparation for use, ensure the following is in place

Environment

- Vault environment
 - Supported version: Any. This has been tested with vault version 12.2
- PowerShell
 - Version 7, tested on Mac and Windows 10 Professional
- Licensing
 - No additional licensing is required

Knowledge

The end user should have the following pre-requisite knowledge

- Fundamental knowledge of the CyberArk PAS solution
- Knowledge of PowerShell scripting is helpful

Required Access

The end user should have the following access

- Administrative privileges on the vault
- A separate user should be created for access to the vault with the API

Installation

The script has two parts

1. GetPendingAccounts.ps1 – this is the actual script
2. GetPendingAccountsEnv.ps1 – this script contains global variables

The two scripts must reside in the same folder / directory. The scripts can be placed in any folder or directory.

Operational Overview

Syntax

GetPendingAccounts.ps1

Arguments

No arguments are required

Database / Vault Impact

This script is a 'read only' script that will not modify the database.

Operation

The script uses the 'DiscoveredAccounts' API call (more information is located [here](#)) to get all of the accounts that are in the <CPM>_pending safe, where <CPM> is the name of the PasswordManager user.

The initial output from the API call is a JSON file, which is then formatted into a CSV file. The data, before conversion contains the following fields:\

Id: Unique ID assigned to the account

Name: The formatted username, used as an identifier within the vault

userName: This is the account's username

address: This is the target server

discoveryDateTime: This is the time (in seconds since epoch) that the account was discovered

accountEnabled: Flag to indicate if the account is enabled

osGroups: List of the groups that the user belongs to

platformType: Based on the type of account, assigns a default platform

lastLogonDateTime: Time of last login (in seconds since epoch)

lastPasswordSetDateTime: Time of last password change (in seconds since epoch)

passwordNeverExpires: Flag to show if the password will expire

osVersion: Detected operating system

privileged: Flag to show if the account is privileged

userDisplayName: User name

description: Description

passwordExpirationDateTime: Time of password expiration (in seconds since epoch)

osFamily: This will usually be 'Server'

additionalProperties: Contains information about the discovery

platformTypeAccountProperties: In Windows, this will be a unique SID and in *NIX this will be the UID

The CSV file will contain the ID, Username, Name, Address, Description, Password Age (in days) and Privileged fields. Addition of other fields requires the script to be modified.

Implementation

Preparation

1. Create a vault user with password authentication that has the same privileges as the default 'administrator' user. This user should be placed in the 'Vault Admins' group.

Operation

1. Accept the disclaimer

2. Review the process and press 'c' to continue
3. When prompted, enter the full URL for a PVWA
4. Enter the username and password when required
5. You can filter the search by specifying
 - a. Platform Types (NOTE ** Only one platform filter is allowed in the search **)
 - i. Windows Server
 - ii. Windows Desktop
 - iii. Windows Domain
 - iv. UNIX
 - v. UNIX SSH Keys
 - vi. AWS
 - vii. AWS Key Access
 - viii. Azure Password Management
 - b. Privileged Status
 - i. You can filter by privileged or non-privileged accounts. The default is no filter
 - c. You can filter by the account's status as taken from the target host (Enabled or disabled)
6. When prompted for the filename, either accept the default './pendingAccounts.csv' (This will be created in the same folder as the script) or input another filename. Ensure that a valid path is entered (Hint: ./ can be used to store the file in the current directory)

Miscellaneous

Data Output

The output from the API call contains:

Default Values

The GetPendingAccountsEnv.ps1 contains global variables for use during operation. The variables that can be changed without adverse effect are

- **\$retLimit:** By default, the 'DiscoveredAccounts' call will only return 100 accounts. The limit in the script is set to 1000 which should meet the needs of most end users.
- **\$defCSVFN:** This is the default file name for the output CSV file. This can be changed to meet needs as appropriate. If specifying a path, it is recommended to include a full path.
- **\$csvHeader:** This is the header that precedes the CSV file. The field names within the header can be changed, but the order cannot be changed and extra fields cannot be added without modifying the script.

**** NOTE ** If variables are changed, please remember to save the file before running the script**

Known Issues

- The logoff sequence returns an error, but this is due to the session being disconnected beforehand
- A '-' in the Password Age field denotes an account where the date of the last password change, and defaults to the number of seconds since 1/1/1970 (Incidentally, this is the beginning of UNIX time)