PASSWORD RBL

PRBL-CBLMANAGEMENT.PS1 - INSTRUCTIONS FOR USE

Summary

The `prbl-cblmanagement.ps1` file is a Windows PowerShell script written to interact with the Password RBL Custom Blacklist Management API. This script implements all the API functions and can be run from anywhere (not just the systems that use the Source IPs provided to Password RBL during registration). There are no third-party libraries or plugins. This code uses all natively provided functionality built in to PowerShell.

Requirements

The prbl-cblmanagement.ps1 script does require Windows PowerShell and should be run from a system whose OS is Windows 7 or Server 2008 R2 or later to ensure complete compatibility. By default, the supplied passwords are interactively hashed 30,000 times, so the faster the computer's processor, the faster the program can submit custom blacklist entries. However, it is important to note that connections to the webservice API are throttled to an average of 4 API calls per second.

Check PowerShell Execution Policy

Windows PowerShell has a built-in security model that can deny the ability to run scripts. You may need to update your PowerShell Execution Policy to allow this script to run. To do so, utilize the built-in `Set-ExecutionPolicy` cmdlet. Password RBL recommends the policy be set to "RemoteSigned" to gain the benefit of requiring remotely located scripts to be signed, but local scripts to be allowed. You may need to run the PowerShell window with elevated permissions (aka Run As Administrator) in order to change the system level Execution Policy. This would only be necessary to change the Execution Policy. The `prbl-cblmanagement.ps1` script does not required elevated permissions. An example command is as follows:

```
Set-ExecutionPolicy -RemoteSigned
```

Create a text file of Passwords

The `prbl-cblmanagement.ps1` script takes a text file as an argument when you are adding or removing entries from the custom blacklist. This file should be a simple, unformatted, text file with one cleartext password per line.

Do not enclose each password in double-quotes or any other delimiting characters.

Using the Script

Summary

```
prbl-cblmanagement.ps1  MODE} <BlacklistID>  [PATH]
```

Arguments

MODE

The mode argument is the verb that dictates what `prbl-cblmanagement.ps1` will do during this execution. The options are Quota, Count, Add, Delete, or Empty.

Quota

This mode will connect to the API and return the maximum number of allowed custom blacklist entries for the specified BlacklistID.

Count

This mode will connect to the API and return the current number of custom blacklist entries.

Note: Password RBL supports multiple hashing algorithms. Currently, PBKDF2 and SHA256 are supported, but others may be added in the future. You only need to add hashes of the passwords in your input file using the algorithm that you plan to use when you query the API during password change events. However, it is not detrimental to add both types. So, for ease of use, Password RBL recommends you add passwords hashed with both algorithms. The Count method will return maximum number of custom blacklist entries across each hash type.

Add

The Add action uses the provided input file, hashes each password, and injects each resulting hash value into the custom blacklist, identified by the BlacklistID.

Delete

The Delete action uses the provided input file, hashes each password, and removes each resulting hash value from the custom blacklist, identified by the BlacklistID.

Empty

The Empty action deletes all entries (of any hash type) from the custom blacklist in a single command.

BlacklistID

The BlacklistID is always required and identifies what custom blacklist will be manipulated by the script.

Path

The Path argument is only required when the mode is either Add or Delete. This argument contains a path to an unformatted text file of cleartext passwords to either Add or Delete from the custom blacklist identified by the accompanied BlacklistID argument. See the appendix for suggestions on making a good list of custom passwords to block.

Examples

Adding entries to the blacklist

```
PS C:\> prbl-cblmanagment.ps1 ADD 12345678901234567890123456789012 C:\pwds4prbl.txt
```

Checking a blacklist's quota

```
PS C:\> prbl-cblmanagment.ps1 QUOTA 12345678901234567890123456789012
```

Check how many entries are in the blacklist

```
PS C:\> prbl-cblmanagment.ps1 COUNT 12345678901234567890123456789012
```

Remove entries from the custom blacklist

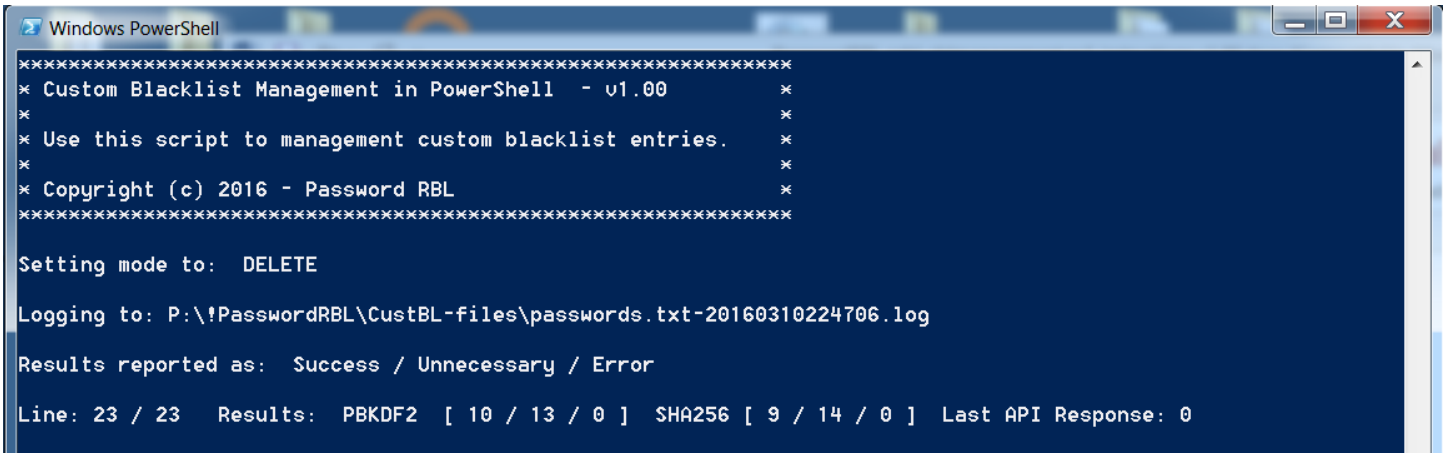
```
PS C:\> prbl-cblmanagment.ps1 DELETE 12345678901234567890123456789012 C:\pwds2del.txt
```

Remove all entries from the custom blacklist

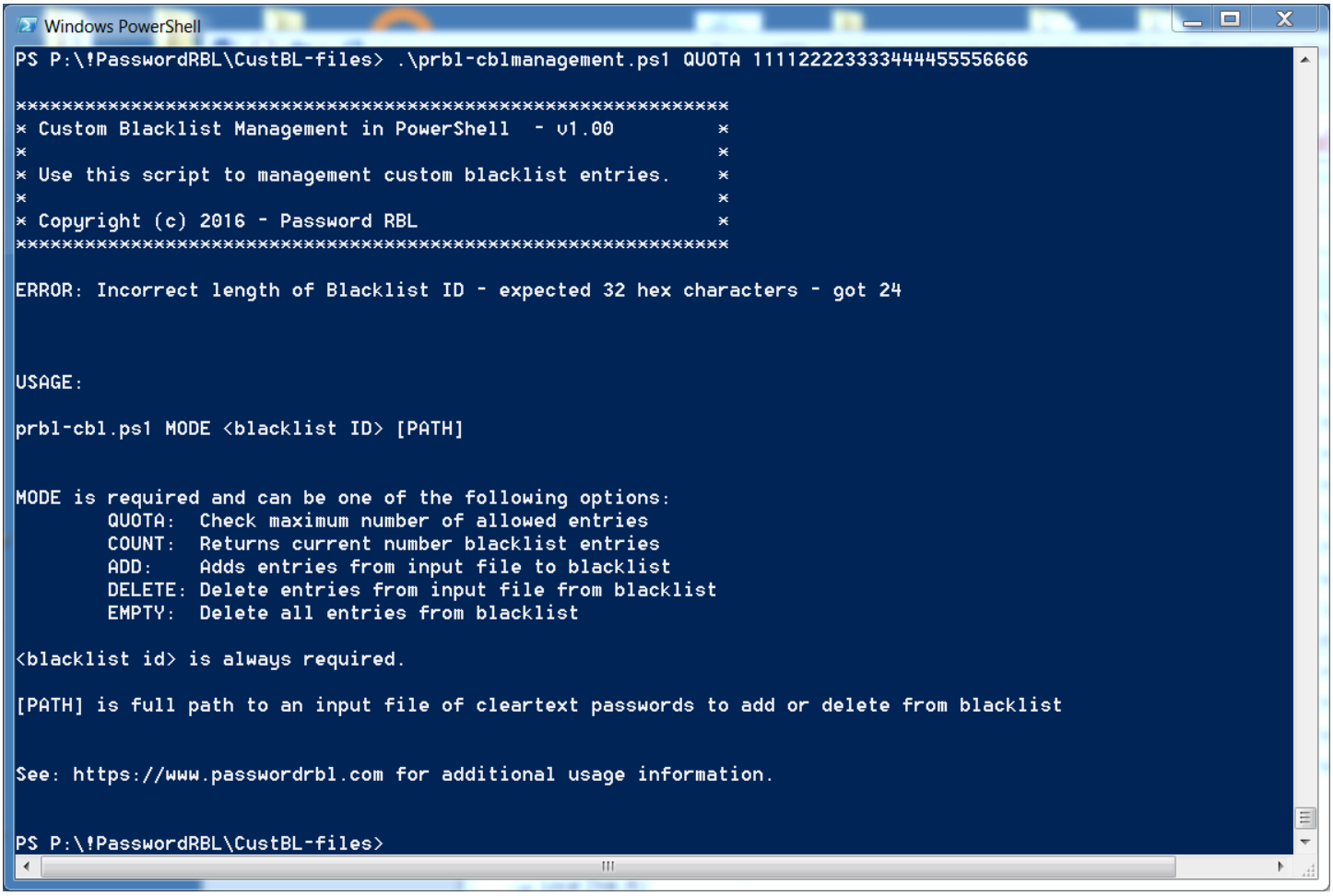
```
PS C:\> prbl-cblmanagment.ps1 EMPTY 12345678901234567890123456789012
```

Output from Script

Good Results



Bad Results



Logging

When the MODE is set to Add or Delete, the `prbl-cblmanagement.ps1` script logs the each individual API call to a file to aid in troubleshooting in case of errors. This file will attempt to be put in the exact same folder as the input file. If this fails [likely due to disk permissions], then the script places this file in the home folder of the user running the script. The name of the log file will be the same name as the input file with the current date and time appended to the end of the file name.

Appendix: Tips for Custom Blacklist entries

Below you will find some tips for what types of passwords to add into your custom blacklist. These tips are based on work done by many individuals and academics, both in directed studies as well as analysis of real credential data following public data breaches.

Each Entry is [Case] Specific

Password RBL's database of bad passwords enumerates all the permutations of known bad passwords (as well as passwords caught by honeypots and obtained from public data breaches). The custom blacklist functionality works exactly the same way. This means you must add all permutations of a password you want to block. For example, you need to enter both: `example123` and `Example123`.

Incrementing Serial Numbers Follow Base Password

It is common that end-users will add an increasing number to the end of their unchanging "base" password. This practice stems from a system or company's password policy that requires end-user password changes on a set frequency (monthly, quarterly, etc.). When this happens, end-users tend to pick one password, and just change the number at the end of their chosen password. You may want to add permutations of commonly use passwords at your company with serial numbers appended to the end. For example, `CompanyPassword1`, `CompanyPassword2`, `CompanyPassword3`, etc., etc.

Company Name and other Public Data

Many employees will choose passwords that are associated with the place of work. This helps them remember the password, but unfortunately, much of this data is publically available. You probably want to block the use of these passwords, and maybe also block them with appended numbers at the end. Common choices are:

- The company name
- The company's address or the address of the site where they work
- The company's phone number
- Company Name with address or phone number appended
- Company Name with the year appended
- Company motto or slogan

Previously Used Passwords

Another good set of passwords to add to your custom password blacklist would be any previously used shared passwords. This is especially true if these passwords were used with accounts that have elevated permissions. Such practice is common amongst IT departments as IT workers have significantly higher number of credentials to remember compared to the typical employee. Adding these shared [admin] passwords to your custom password blacklist is especially important when an IT worker leaves the company. You can then enforce that a password change happen and know that accounts are not using a password that a [now] non-employee knows and could easily guess to gain unauthorized access after they leave the company.