



## Threat Response “as is” PowerShell

### Change Password At Logon

**Revision Number** 0.2

**Date** July 9, 2018

**Prepared By** Stephen Sullivan  
Senior Sales Engineer  
Proofpoint, Inc.  
(770) 207-1536  
[SSullivan@proofpoint.com](mailto:SSullivan@proofpoint.com)



Table of Contents

Disclaimer..... 3

Problem Statement ..... 3

Project Objective ..... 4

Solution Summary..... 4

Basic Requirements ..... 5

Configuration ..... 5

Exception Handling and Logging ..... 11



## Disclaimer

These instructions and associated PowerShell script are provided “as is”. Every effort was made to adhere to PowerShell best practices. Please keep in mind, this is a “Script” and not an “Application”. As such, logging is limited to script actions and assumes underlying PowerShell, Windows Authentication and associated modules/methods are configured and functioning correctly.

Lastly, this script and document only covers Email related remediation and therefore should not be treated as an exhaustive solution. Endpoint, Proxy, Firewall and a number of other tools can and should be used.

## Problem Statement

### Primary:

Proofpoint Targeted Attack Protection (TAP) tracks and reports Domain User(s) who have previously “clicked” and were “allowed” to continue to a URL that has since been deemed malicious. This is usually the result of URLs that have been “weaponized” post-delivery. While “Click” alerts do not equal infection, Many organization force uses to change their password as a protective measure.

It is important to note that all future “clicks” on malicious URLs will be blocked and the message no longer poses a threat. As a matter of hygiene and to prevent users from reporting messages already marked malicious, it is a good idea to remove these messages from the inbox with Threat Response Auto-Pull (TRAP)

### Secondary (optional-highest level of protection):

Proofpoint TAP tracks and Reports on Attachments with embedded URLs. Because these URLs exist in Attachment, the URL links cannot be rewritten. These URLs are also susceptible to “weaponization” post-delivery. Because the URLs are not rewritten, Proofpoint cannot report if the Domain User as accessed the URL. Due to the lack of visibility, some Organization may choose to force Domain recipients to change their password as well.

Because the malicious URLs have not been rewritten, these messages still pose a threat and need to be removed from the Domain Recipient’s Inbox. Threat Response Auto-Pull (TRAP) is the fastest way to respond to these threats.



## Project Objective

To leverage Targeted Attack Protection (TAP), Threat Response Auto-pull (TRAP) and Windows PowerShell to automate the removal of email threats for Domain Recipient mailboxes and force Domain Users who have been potentially exposed to malicious content to change their password at next login.

## Solution Summary

Targeted Attack Protection (TAP) provides alerting and tracking of User Mailbox threats. TAP support authenticated API access to alert, campaign and forensic detail.

Proofpoint Threat Response (PTR) Auto-Pull (TRAP) leverages the TAP APIs to manage alerts and provides many options for remediation. In addition to extensive built-in functionality (e.g. TRAP), PTR also provides a REST API.

For this use case we will only be discussing Auto-Pull and the PTR API.

Auto-Pull will be used to move malicious Emails from Recipients' mailboxes to a quarantine in a secured mailbox. PTR quarantines the original email, any forwarded emails and all email of Distribution List members.

Additional details regarding TRAP installation and configuration can be found in the Threat Response portal, which is accessible directly from the PTR console, or via the link below:

<https://ptr-docs.proofpoint.com/ptr-guides/ptr-about/>

In addition to Auto-Pull, PTR can be configured to automatically add users to a "List". Members of a list can be retrieved via a secure REST API web GET request to the PTR server. Members can also be deleted with a secure web DELETE request. Additional detail regarding the PTR API can be access via the PTR Portal from a licensed PTR Console.

Windows Task Scheduler is used to execute a PowerShell script every *n* minutes. This script uses a web GET request to retrieve all members of the configured list, validates the account against AD users and verifies the AD User meets requirements. Validated AD Users will be forced to change their password at next login. A web DELETE request removes the user from the configured list.

Script log are stored in a configured path on the PowerShell hosting the Scheduled task. Updates are also shown in the PTR Console.



## Basic Requirements

- Account with 'Administrator' access to Proofpoint Targeted Attack Protection (TAP)
- Licensed version of Proofpoint Threat Response Auto-Pull (TRAP) installed
  - <https://ptr-docs.proofpoint.com/trap-guides/trap-installation/>
- An Active-Directory Account with a mailbox and with one of the Exchange Roles:
  - "Full Access" permission
  - Application Impersonation role
- An Active-Directory Account with the following permission/capabilities
  - Domain User to Connect and Query AD/LDAP
  - Modify AD User Properties
    - Passwordneverexpires
    - ChangePasswordAtLogon (a.k.a. Pwdlastset)
    - CannotChangePassword
  - Run a PowerShell script in Windows Task Scheduler
  - Write Access to the Script 'Log' path (e.g. C:\Scripts\)
- Windows Domain computer with PowerShell and PC Active-Directory Module
- The provided PowerShell Script

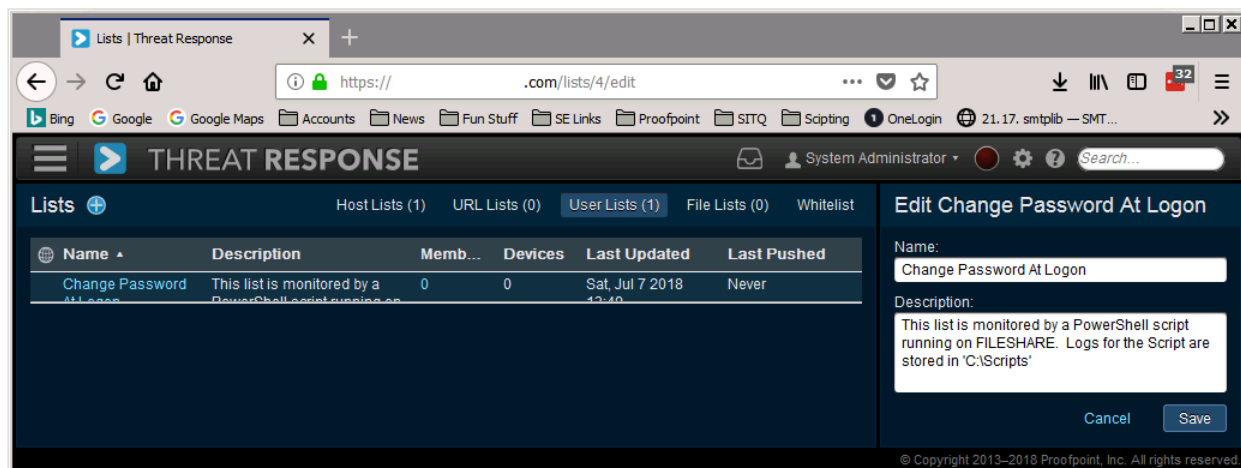
## Configuration

TRAP installation and configuration is well documented, so I will fore go any repetition and assume TRAP is up and running. I recommend going through the TRAP set up first. It will get you acclimated to the PTR console and introduce some concepts we will use (e.g. Match Conditions).

### Step 1: Configure a 'User List' in Threat Response

When creating the 'User List', **make a note of the List number (aka. Id)** in the URL. We will use this number for our web GET and DELETE requests. In the example the List ID is '4' (figure 1.1).

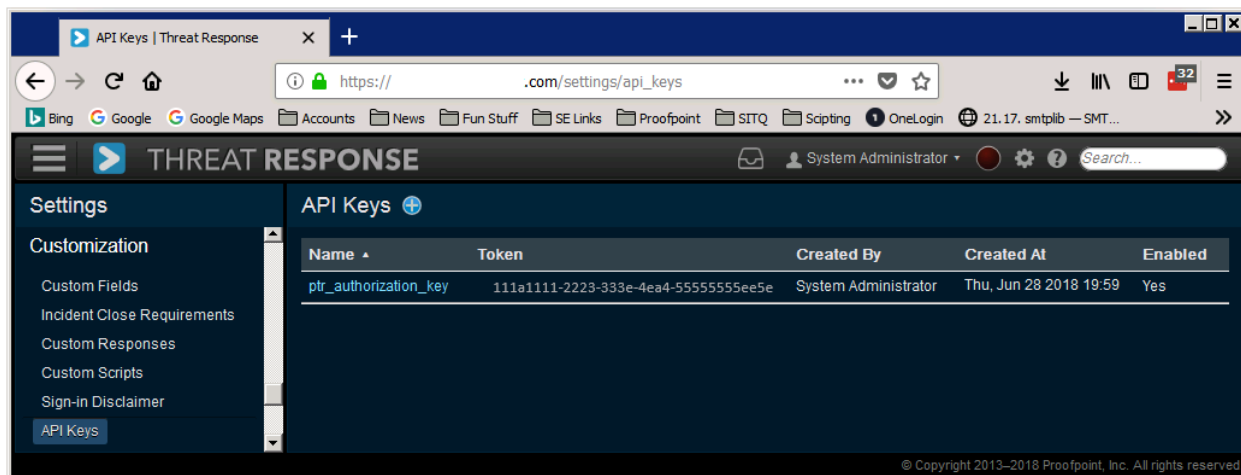




(figure 1.1)

## Step 2: Generate a Threat-Response API key

Logon to the Proofpoint Threat Response console. Access PTR settings. Under Customization and API Keys generate a new key and make a note of the Token (figure 2.1)

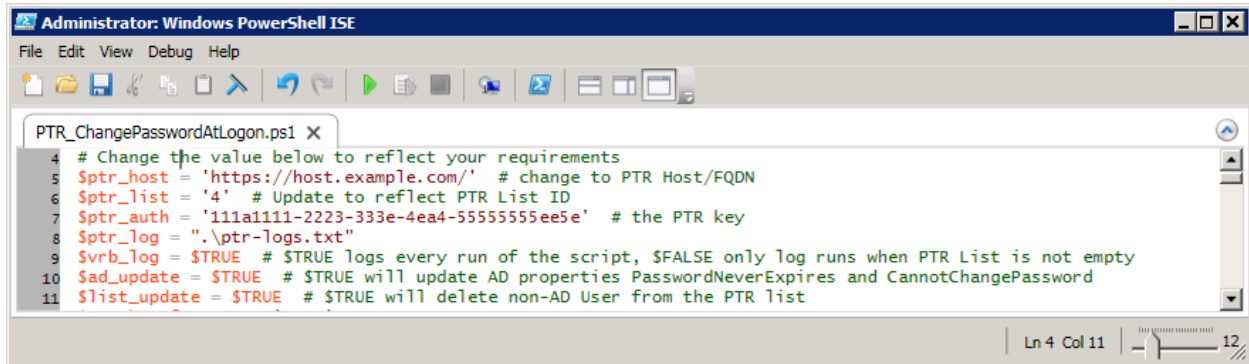


(figure 2.1)

## Step 3: Configure environment variables in the PowerShell Script

Logon to the Domain system that will host the Script as the user account that will run the Script in Windows Task Schedule. Open the provided Script in Windows PowerShell ISE. Update the variables as directed in the comments (figure 3.1)



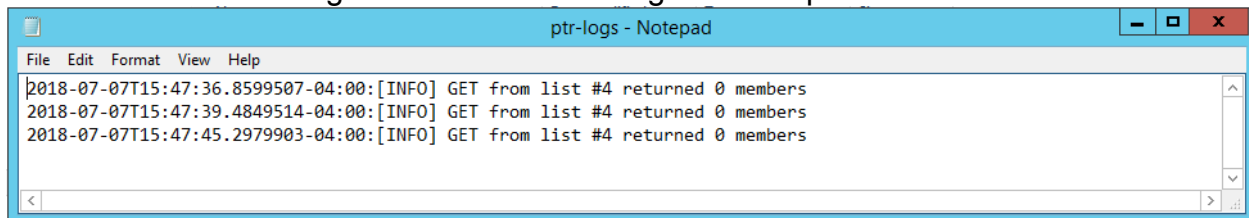


(figure 3.1)

#### Step 4: Test the Script in Windows PowerShell ISE

At this point you should be able to run the PowerShell script without receiving an error. Because you are running the Script for PS ISE, the log is created in the same directory as the Script. Open the log and verify it is working (figure 4.1).

\* Because the Matching Rule is disabled the log should report '0' members

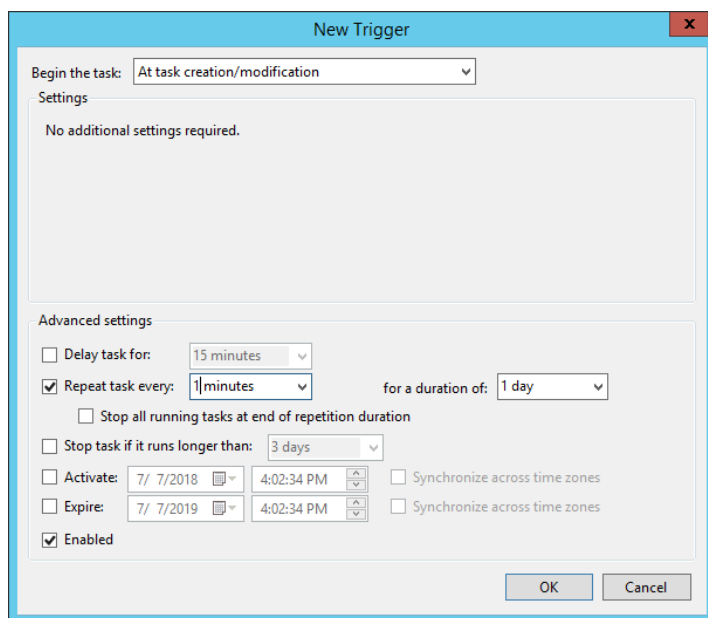


(figure 4.1)

#### Step 5: Create a 'Scheduled Task' to run the PS Script

Open 'Windows Task Scheduler' and select 'Create Task' (not 'Create Basic Task') under actions.

- General –
  - Configure task to run as the 'PTR User'
  - Select 'Run with highest privileges'
- Triggers –
  - Configure as shown (figure 5.1)
  - The default minimum is '5 Minutes'. Click on the '5' and change to '1'



(figure 5.1)

- Action –
  - Update the parameters below to reflect Script location. Use 'Start in' to determine where log file will be written.
  - Program/Script:
    - 'PowerShell.exe'
  - Add Args:
    - '-ExecutionPolicy Bypass C:\Scripts\PTR\_ChangePasswordAtLogon.ps1 -RunType \$true'
  - Start in:
    - 'C:\Scripts\'
- Settings –
  - Update 'Stop the task if it runs longer than:' to '1 hour'

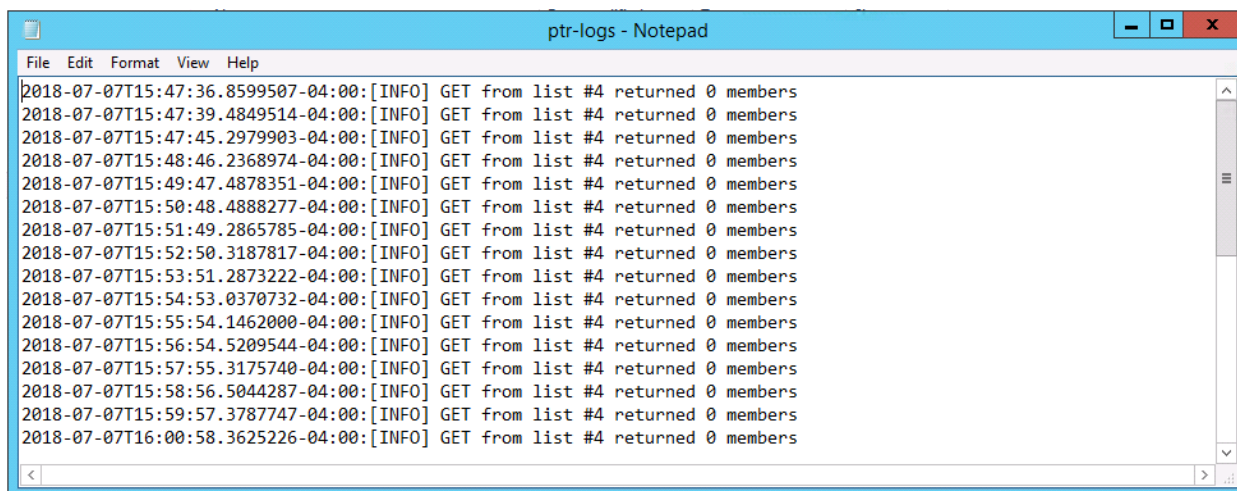
#### Step 6: Good time to Check the log and update '\$vrb\_log'

Open the PTR Log. It should be in the 'Start in:' folder you specified during task creation. When \$vrb\_log is set to '\$TRUE' every run of the script is logged (figure 6.1)

I recommend changing the \$vrb\_log variable to false in the script to limit logs to only when 1 or more members is found in the PTR List (figure 6.2)

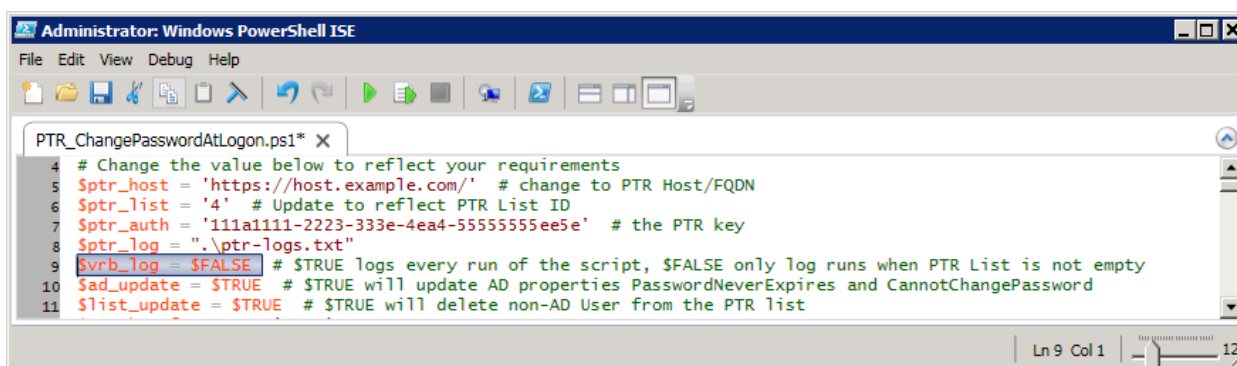






```
File Edit Format View Help
2018-07-07T15:47:36.8599507-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:47:39.4849514-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:47:45.2979903-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:48:46.2368974-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:49:47.4878351-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:50:48.4888277-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:51:49.2865785-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:52:50.3187817-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:53:51.2873222-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:54:53.0370732-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:55:54.1462000-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:56:54.5209544-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:57:55.3175740-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:58:56.5044287-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T15:59:57.3787747-04:00:[INFO] GET from list #4 returned 0 members
2018-07-07T16:00:58.3625226-04:00:[INFO] GET from list #4 returned 0 members
```

(figure 6.1)



```
Administrator: Windows PowerShell ISE
File Edit View Debug Help
PTR_ChangePasswordAtLogon.ps1* X
4 # Change the value below to reflect your requirements
5 $ptr_host = 'https://host.example.com/' # change to PTR Host/FQDN
6 $ptr_list = '4' # Update to reflect PTR List ID
7 $ptr_auth = '111a1111-2223-333e-4ea4-55555555ee5e' # the PTR key
8 $ptr_log = ".\ptr-logs.txt"
9 $vrb_log = $FALSE # $TRUE logs every run of the script, $FALSE only log runs when PTR List is not empty
10 $ad_update = $TRUE # $TRUE will update AD properties PasswordNeverExpires and CannotChangePassword
11 $list_update = $TRUE # $TRUE will delete non-AD User from the PTR list
```

(figure 6.2)

### Step 7: Create a 'Match Rule' to add Users to the List

Go to the TAP Source created during the TRAP set up. Create a new 'Match Rule' to match the Primary and Secondary objectives (figure 7.1).

- **Primary** - Force user who “click” on Phish, Malware and Imposter links to change password at next login.
  - Alert Type: 'Permitted Clicks'
- **Secondary** – Force users who have received attachments threats and unprotected URLs to change password at next login.
  - Alert Type: 'Delivered Attachment Threats'
  - Alert Type: 'Unprotected URL Threats'



(figure 7.1)

## Step 8: Validate functionality

You should now be able to manually add Users to the PTR List to validate functionality. Inside the PTR Console, within approximately ~1 minute you will see the User has been removed from the List. This means the User's AD property of 'ChangePasswordAtLogon' has been set to \$TRUE (figure 8.1).

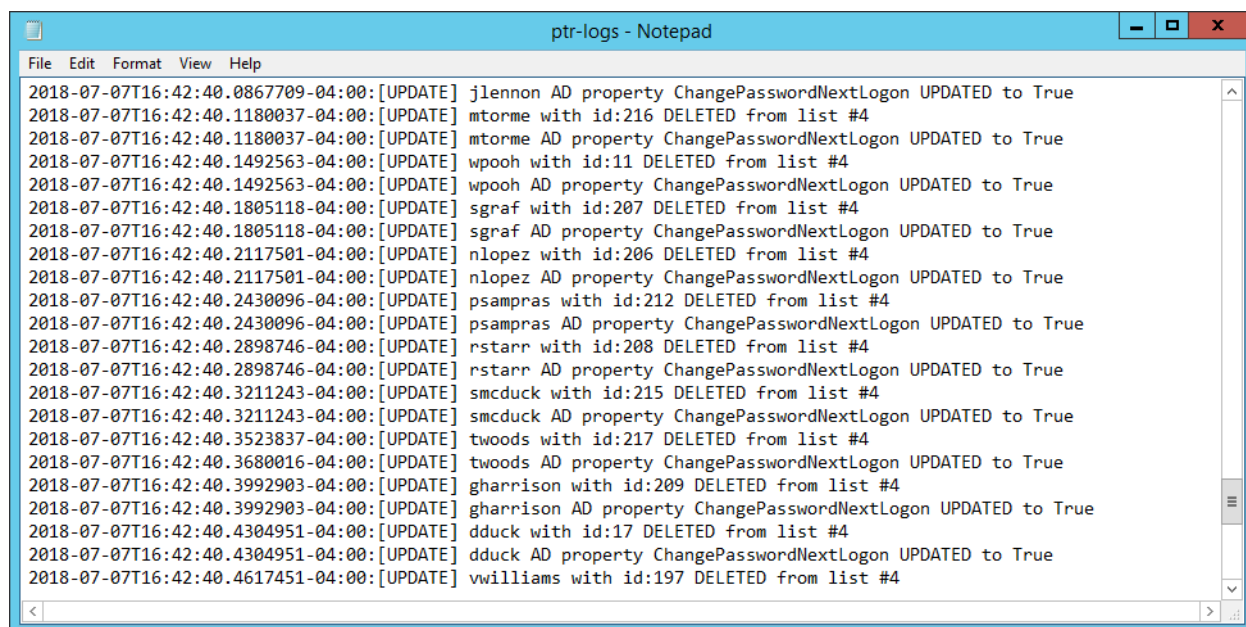
Check the PS Script logs for additional detail (figure 8.2)

Name	Description	Memb...	Devices	Last Updated	Last Pushed
Change Password At Logon	This list is monitored by a PowerShell script running on FILESHARE. Logs for the Script are stored in 'C:\Scripts'	0	0	Sat, Jul 7 2018 16:34	Never

**Recent Changes**

User	Action	Time
Jimmy Olsen (j...)	Removed by System	3 minutes ago
Lois Lane (llane)	Removed by System	3 minutes ago
Arnold Palmer ...	Removed by System	3 minutes ago
Mickey Mouse (...)	Removed by System	3 minutes ago
Winnie Pooh (...)	Removed by System	3 minutes ago

(figure 8.1)



```

ptr-logs - Notepad
File Edit Format View Help
2018-07-07T16:42:40.0867709-04:00:[UPDATE] jlennon AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.1180037-04:00:[UPDATE] mtorme with id:216 DELETED from list #4
2018-07-07T16:42:40.1180037-04:00:[UPDATE] mtorme AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.1492563-04:00:[UPDATE] wpooh with id:11 DELETED from list #4
2018-07-07T16:42:40.1492563-04:00:[UPDATE] wpooh AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.1805118-04:00:[UPDATE] sgraf with id:207 DELETED from list #4
2018-07-07T16:42:40.1805118-04:00:[UPDATE] sgraf AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.2117501-04:00:[UPDATE] nlopez with id:206 DELETED from list #4
2018-07-07T16:42:40.2117501-04:00:[UPDATE] nlopez AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.2430096-04:00:[UPDATE] psampras with id:212 DELETED from list #4
2018-07-07T16:42:40.2430096-04:00:[UPDATE] psampras AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.2898746-04:00:[UPDATE] rstarr with id:208 DELETED from list #4
2018-07-07T16:42:40.2898746-04:00:[UPDATE] rstarr AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.3211243-04:00:[UPDATE] smcduck with id:215 DELETED from list #4
2018-07-07T16:42:40.3211243-04:00:[UPDATE] smcduck AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.3523837-04:00:[UPDATE] twoods with id:217 DELETED from list #4
2018-07-07T16:42:40.3523837-04:00:[UPDATE] twoods AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.3680016-04:00:[UPDATE] gharrison with id:209 DELETED from list #4
2018-07-07T16:42:40.3680016-04:00:[UPDATE] gharrison AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.3992903-04:00:[UPDATE] dduck with id:17 DELETED from list #4
2018-07-07T16:42:40.3992903-04:00:[UPDATE] dduck AD property ChangePasswordNextLogon UPDATED to True
2018-07-07T16:42:40.4304951-04:00:[UPDATE] vwilliams with id:197 DELETED from list #4
2018-07-07T16:42:40.4304951-04:00:[UPDATE] vwilliams AD property ChangePasswordNextLogon UPDATED to True

```

(figure 8.2)

## Exception Handling and Logging

There are three general exceptions to can be handled by this script. Below are details on each

1. A name added to the User List in Threat response doesn't exist in Active Directory

Because PTR correlate and does not validate AD Users, there is always a chance and admin or alert could result in a non-AD Account name being added to the User List.

Default: @list\_update = \$TRUE

To prevent repeated errors, non-AD Users are removed from the list.

```

$ad_update = $TRUE # $TRUE will update AD properties PasswordNeverExpires and CannotChangePassword
$list_update = $TRUE # $TRUE will delete non-AD User from the PTR list

```

Log Sample:

```

2018-07-07T16:32:29.9564252-04:00:[ERROR] www.aserimovbil.com does not exist in Active Directory
2018-07-07T16:32:29.9878167-04:00:[UPDATE] www.aserimovbil.com with id:15 DELETED from list #4

```



2. The AD User account is set to 'User Cannot Change Password'  
(and)
3. The AD User password is set to 'Never Expire'

Default: @ad\_update = \$TRUE

To prevent repeated errors and force 'ChangePasswordAtLogon' both AD Properties will be updated to \$FALSE.

```
$ad_update = $TRUE # $TRUE will update AD properties PasswordNeverExpires and CannotChangePassword  
$list_update = $TRUE # $TRUE will delete non-AD User from the PTR list
```

## Log Sample:

2018-07-07T16:32:30.0657694-04:00:[INFO] mtorme AD property PasswordNeverExpires UPDATED to FALSE

2018-07-07T16:32:30.0813935-04:00:[UPDATE] mtorme with id:216 DELETED from list #4

2018-07-07T16:32:30.0813935-04:00:[UPDATE] mtorme AD property ChangePasswordNextLogon UPDATED to True

