Managing Lost USB Devices

Peter Stone
Technology Manager, Library, University of Waikato

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Purpose

- Enhance user experience returning property promptly
- Reduce staff time processing lost USB devices
- Largely eliminate the need for reviewing the USB content

Assumptions

- Windows operating system.
- PowerShell – **Set-ExecutionPolicy** to **RemoteSigned** via Group Policy
- Using WMI class of **Win32_ComputerSystem** on the local machine for user details
- Using WMI class of **Win32_DiskDrive** for surfacing USB details

WMI USB Device Data Fields (Win32_ComputerSystem)

- **DeviceID** Mostly unique by manufacturer
- **Caption** Combined with DeviceID seems to be a valid key
- **SerialNumber** Only used if the file has more than 2 characters
- **MediaType** is always “Removable Media” (so is not being used)
- **Size** (in bytes) Returns different values for different versions of Windows!

Script Functions

- Single PowerShell script written so it can be run in two ways:
  1. **Detection/Recording**
  2. **Detection/Interrogation** (run using an argument)
- Script interacts with a web API written in PHP that performs the interaction with the Database.

Detection/Recording

Initiated via Group Policy using either "Computer" OnStart or "User" Logon script triggers.

- Once running (as either a Process or a Service) detection is performed at 5 minute intervals.
- The script queries the WMI interface to return the metadata associated with any USB found plugged in.
- WMI data is compared to the contents of a log file (C:\Users\Public\USB.txt) to confirm if the USB has been previously seen on this machine.
- Processing of the USB stops if it has been already “seen” by the script, thereby minimizing the incidence of multiple usernames being associated with it.
- If the device was not previously recorded on the PC, then the USB metadata together with the username, computername, date and time, is recorded in the local file AND sent via API to the database.

Detection/Interrogation

By changing how the script is called, the script can be used to interrogate a USB, to see if it matches records harvested.

- Typically staff will plug in the USB and run the script from a shortcut which calls PowerShell explicitly with two arguments: The first argument is the path to the script itself and the second argument is anything you want - say a “x"
- If there are matching records to the USB in the database, the script will cause the API to open Google Chrome and display them
- If there are no matching records, the script will trigger the API to open Google Chrome and advise this

API

- Written in PHP
- Uses stored procedures to insert or query records – providing a measure of protection from SQL injections attacks
- Presently located at https://library.waikato.ac.nz/usb
- The documentation on the page provides the calling syntax with examples on how to enter the USB data or to compare the USB data to what might already be in the database.

Database

- Single table **MariaDB**
- Could be any suitable Database….

Two Pronged Strategy

- We prefer not have to deal with USB's at all.
- To that end, we have a separate process that employs a small application to control the pc sound and a short script that runs on logoff.

“Remove your USB” Logoff Script

- Triggered in Group Policy on the "User" Logoff script

IF it detects a USB storage device:

- Displays a message on screen
- Turns the volume up
- Plays a short sound bite advising the user to “Remove your USB”
- Turns the volume down

Resources

Source code and instructions for both process are on GitHub and linked to from the API:

https://library.waikato.ac.nz/usb

Acknowledgement

Library Systems Developer, Fred Young who peer reviewed, contributing improvement to the database structure, file distribution and the look and “feel” of the USB API.