# Three Quick Topics

Pi-hole
Windows Subsystem for Linux
CircuitPython

Connie Sieh & Dave Putz
Uniforum Chicago
January 21, 2020

# Pi-hole

- The Pi-hole® is a DNS sinkhole that protects your devices from unwanted content, without installing any client-side software
- It is set up as a DNS server that has a configurable blacklist of sites
- When a DNS request is made to a listed site a fake IP address is returned, which points to a small web server on the Pi-hole system.
- Any requests made to the fake IP address just return a tiny blank image.
- So, your web browser (or other software) thinks it has received an ad, but just displays an empty space.

# Pi-hole

- Prerequisites
  - 512 MB Ram, 52MB disk space
  - A supported operating system
    - Raspbian
    - Ubuntu
    - Debian
    - Fedora
    - CentOS
  - A static IP address for your network

# Pi-hole

- Installation
  - There are 3 methods available to install
    - A one-step automated install using curl
    - A git clone of the Pi-hole repository and running a script it contains
    - A manual download of the installer

    The install will add any required packages that might be missing and ask questions about some configuration options
  - Time to install varies; on a slower wifi it took about 10 minutes
  - Note that the installer will show a randomized admin password; you will want to make a note of it (or use pihole -a -p to set a new one)

# Pi-hole

- Post-Installation
  - Once the install completes successfully Pi-hole will be up and running
  - To use it, just point the DNS server for any device to the IP address of the Pi-hole system
    - If you have control over a dhcp server being used you can set the default DNS server to your Pi-hole
      - Example for dhcpd.conf:

```
option domain-name-servers xxx.xxx.xxx.xxx
```

    - Pi-hole also comes with a built-in dhcp server

# Pi-hole

- Administration
  - If you want to use the default blacklist, no more work is needed
  - There is a web-based admin tool available at the Pi-hole IP address
    - xxx.xxx.xxx.xxx/admin
  - You can get a graphical view of DNS queries – how many were made, how many were blocked, how many clients, etc.
  - There are many options available (such as disabling Pi-hole for a short time so you can see those ads)
  - You can add domains to the blacklist or whitelist
  -

# Pi-hole

- Demo Hardware
  - Raspberry Pi Zero W
  - Sandisk 32GB micro sd
  - Raspbian Lite Stretch
    - Wifi enabled
    - USB serial “gadget” enabled

# Pi-hole

- Demos
  - Install Pi-hole
    - wget -O basic-install.sh https://install.pi-hole.net
    - sudo bash basic-install.sh
  - Point client DNS to Pi-hole ipaddress
  - Admin page
  - Timing of a site with/without Pi-hole
    - cnet.com
    - espn.com

# Pi-hole

- References and guides
  - Pi-hole install guide - https://github.com/pi-hole/pi-hole/#one-step-automated-install
  - Raspbian download https://www.raspberrypi.org/downloads/raspbian/
  - A Raspberry Pi Zero W needs some setup to run with a USB console https://www.tal.org/tutorials/raspberry-pi-zero-usb-serial-console
  - Using Pi-hole with a VPN to provide remote access https://docs.pi-hole.net/guides/vpn/overview/

# Windows Subsystem for Linux

- The Windows Subsystem for Linux (WSL) is a Windows 10 feature that enables you to run native Linux command line tools directly on Windows, alongside your traditional Windows desktop
  - History
  - Installation
  - Use
  - Limitations

# Windows Subsystem for Linux

- HISTORY
  - Microsoft initially tried to be “Unix-y” by providing a Posix subsystem in Windows NT
    - This was done to meet DoD Orange Book specs
    - The POSIX subsystem supported POSIX.1 spec but provided no shells or UNIX-like environment of any kind.
    - Eliminated in Windows XP and Windows Server 2003
  - MS also bought a product named Interix which became a core operating system component, rebranded as “Subsystem for UNIX Applications” (SUA).
  - Not much use or acceptance for either of these products

# Windows Subsystem for Linux

- In Windows 7 Microsoft promoted the use of virtual machines to run Linux distributions
  - These have the same issues as all fully-virtualized environments
    - Less efficient use of hardware
    - Dedicated resources limit overall usage
- WSL was released for 64-bit Windows 10 (version 1607+)
  - It lets you run unmodified Linux binaries directly on Windows 10
  - Command line tools were the focus
  - Provides a “translation” of Linux systems calls to Windows NT kernel calls

# Windows Subsystem for Linux

- INSTALL
  - Via “enable optional feature” setup  OR
  - Via a Admin enabled Powershell
  - `Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`
    - Some necessary software will download, and the WSL subsystem will be enabled after you reboot.
- Install Linux Distro
  - Current available distros are
    - Ubuntu …
    - Kali Linux
    - Debian
    - Alpine WSL
    - SUSE …
    - Pengwin ...
    - Fedora Remix for WSL
  - You can also download & install without Microsoft Store via curl https://docs.microsoft.com/en-us/windows/wsl/install-manual

# Windows Subsystem for Linux

- Install Linux Distro via Microsoft Store
  - Select “get”
- Install Linux Distro via  powershell
  - “curl.exe -L -o ubuntu-1804.app https://aka.ms/wsl-ubuntu-1804”
  - Add-AppxPackage .\ubuntu-1804.appx

# Windows Subsystem for Linux

USE

- List WSL distro
  - “wsl -l --all”  # lists all wsl distros
- Start distro
  - Ubuntu-1804  # starts wsl with specified distro
  - “wsl -d Ubuntu-1804”
  - Select Distro from Start Menu”
- Run linux commands from Windows
  - “wsl <command>”
  - Pipes and redirection works

# Windows Subsystem for Linux

FILESYSTEM ACCESS

- Linux filesystem available in Windows
- Linux “/” filesystem available via “9P” filesystem (1903)
  - https://devblogs.microsoft.com/commandline/whats-new-for-wsl-in-windows-10-version-1903/
  - “\\wsl$\<distro name>\”
- Windows “c:” available in Linux at

  *“/mnt/c/”*
- DO NOT TOUCH APPDATA folder
  - BAD things will happen

# Windows Subsystem for Linux

- WSL 1 runs Linux binaries by partly implementing a Linux API compatibility layer in the Windows kernel , thus some programs will not work because they need the missing system calls
- Since a compatibility layer is used some operations can be slower than native Linux such as disk reads/writes
- WSL 2 Announced May 2019 , First “test” release June 2019, Expected Release 20H1
- WSL 2 uses a Microsoft provided Linux kernel running in a lightweight VM to provide better compatibility with native Linux installations
- Install WSL 2
  - Enable the 'Virtual Machine Platform' optional component via a Admin enabled Powershell
  - “dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart”
  - Set your distro version using a command line:
    - `“wsl --set-version <Distro> 2”`
  - Check the version for each distro:
    - `“wsl --list --verbose”`

# Windows Subsystem for Linux

- Demos
  - WSL 1 Install
    - Via Admin enabled Powershell
      - Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
  - WSL Distro Install – Via Powershell
    - curl.exe -L -o ubuntu-1804.app
    - Add-AppxPackage .\ubuntu-1804.appx

# Windows Subsystem for Linux

- Demos
  - WSL  Usage
    - Microsoft Powershell
      - “wsl  ls”
      - “wsl ls |> filenames”
      - “cd \\wsl$\ubuntu-1804”
      - dir
    - Linux
      - “ls”
      - “explorer.exe”
      - ‘notepad.exe”
      - “cd */mnt/*c”
  -

# Windows Subsystem for Linux

- Demos
  - WSL 2 Install
    - Via Admin enabled Powershell
  - dism.exe /online /enable-featur/featurename:VirtualMachinePlatform /all
    - Restart Windows to enable
  - Set your distro version using a command line:
    - “wsl --set-version <Distro> 2”
    - Check the version for each distro:
      - “wsl --list –verbose”
  - WSL 2 Usage
    - “wsl <linux command> 1 # For wsl 1
    - “wsl <linux command> 2 # for wsl 2

# Windows Subsystem for Linux

- Demos

  Build CircuitPython

  - wsl --set-version ubuntu-1804 2
  - “cd ~/git/circuitpython”
  - “make clean”
  - “time make”
  - “wsl –set-version ubuntu-1804 1”
  - “cd ~/git/circuitpython”
  - “make clean”
  - “time make”

# Windows Subsystem for Linux

- Demo Hardware
  - Dell E5440 – I5 43xx, 8GB DDR3, 500GB SSD
  - Windows 10 1903 Education
  - Hyper V
    - Windows 10 Insiders 19035
      - Needed for WSL 2
      - Did not want to replace Windows 10 1903

# Windows Subsystem for Linux

- References and Resources
  - Manage and Configure WSL Distros
    - https://docs.microsoft.com/en-us/windows/wsl/wsl-config
  - Release Notes
    - https://docs.microsoft.com/en-us/windows/wsl/release-notes
  - Page with lots of WSL links
    - https://github.com/sirredbeard/Awesome-WSL
  - WSL Presentation by Mithun Shanbhag
    - https://www.slideshare.net/mithunshanbhag/wsl-windows-subsytem-for-linux
  - Ubuntu WSL Page
    - https://wiki.ubuntu.com/WSL
  - Microsoft Provided WSL Linux Kernel
    - https://github.com/microsoft/WSL2-Linux-Kernel

# CircuitPython

CircuitPython is a Python interpreter that has been implemented as the firmware on some microprocessors. This allows a programmer to develop and run microprocessor applications without the need for external SDKs, compilers, etc. Scripts are run directly by the Python interpreter. We will cover the design philosophy behind it, the community that supports it, and enter and run some examples (live demo!)

# CircuitPython

- History
  - MicroPython was created in 2013 by Damien George
  - Originally supported a Kickstarted STM32 ‘pyboard’
  - His aim was to support standard CPython with as many modules as possible.
  - Constrained by memory sizes (256k of code space, 16k of RAM)
  - MicroPython has been expanded to other architectures (ESP8266, ESP32, PIC, Unix, Windows, etc.)

# CircuitPython

- History
  - MicroPython was forked in 2017 by Adafruit to create CircuitPython
  - Originally done to support SAMD21 chips used by many Adafruit boards
  - Goal: One hardware API regardless of board or chip type
  - Some non-standard Python code from MicroPython was removed
  - A uf2 bootloader was created to simplify getting code on the board
  - Both projects still share code back and forth

# CircuitPython

- How it works
  - Power-up default is to enter a REPL prompt state on the serial/usb port (REPL = read-eval-print-loop; waiting for entry of Python code)
  - A terminal emulator connected to the USB port allows interaction with the system
  - Python commands & scripts can be entered directly
  - The device will also show up as a USB disk drive
    - Access depends on which OS you are running
    - Code, modules, audio & video files can be copied directly to the CIRCUITPY drive
  - Scripts named boot.py or main.py will be run automatically on power-up (no REPL prompt)
  - You can also run arbitrarily named scripts using a REPL command

# CircuitPython

- How it works
  - The most commonly used Python modules are compiled into the core
  - Many modules for specific hardware devices can be installed separately
  - CircuitPython modules have been ported to CPython (i.e. Raspberry Pi, etc.) - the project is named ‘blinka’
    - https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi
    - https://learn.adafruit.com/neopixels-on-raspberry-pi?view=all

# CircuitPython

- CircuitPython Community
  - All the code is on github
    - Core code at https://github.com/adafruit/circuitpython
    - Library code at https://github.com/adafruit/CircuitPython_Community _Bundle
  - All are welcome to contribute; info can be found at: https://github.com/adafruit/circuitpython/blob/master/CONTRIBUTING.md
  - Primary sources of contact/support
    - https://adafru.it/discord
    - https://forums.adafruit.com/viewforum.php?f=60
  - There is a weekly developers call  on discord – search for #circuitpython

# CircuitPython

- Live Demos
  - Demo 1 - booting up and using the REPL prompt
  - Demo 2 - Writing a simple Python script and copying it to the CIRCUITPY disk drive
  - Demo 3 - Lighting up some LED’s on LED string
  - Demo 4 - Sound level dependent LED
  - Demo 5 – Lighting up some LED’s on LED string running on a Raspberry Pi using “blinka” libraries

# CircuitPython

- Demo Hardware
  - Demo 1,2,3,4
    - Adafruit Circuit Playground Express
    - WS2811 50 LED String
    - WS2812 10 LED
  - Demo 5
    - Raspberry Pi 3 B+
    - WS2811 50 LED String
    - Raspbian Buster

# Presentation Materials

- This presentation and CircuitPython demo scripts are all available on github at https://github.com/siehputz/uniforum2020