CCDC PLAYBOOK 2019

Official playbook for AU-ACM Cyber Defense Club

Primary Goals

CCDC is a yearly blue vs red competition.

The objective: maintain up time of an enterprise network and its primary services all while protecting the network from a professional red team.

The enterprise network architecture changes each year, meaning each year team composition should change to meet the challenges presented by the newest iteration of the play field.

Team Composition

Our team composition this year is a close mapping of the enterprise architecture, with Captain and Change Control Officer designated to carry out tasking injections to any of three teams: Windows, Unix, and Firewall administration.

```
graph LR
A(Change Control Officer) --> B(Captain)
C(Windows Team) --> A
D(Linux Team) --> A
E(A/D DNS) --> C
F(Exchange) --> C
G(BIND DNS) --> D
H(E-Commerce) --> D
I(Web Apps) --> D
J(Network Admin) --> A
```

This is a team competition, where coordination and communication are imperative to success.

The Game Plan

Injects and **Incidents** are prioritized. - Injects correspond to tasks received by the Captain. The entire team is responsible for completing each inject. - Incidents correspond to primary services taken down by the red team or *otherwise*

To successfully balance inject completion, incident resolution all while maintaining or improving a security posture, refer to this work flow:

```
graph LR
A(Inject Completion) --> B(Initial Hardening)
B --> C(Harden)
C --> D(Enumerate)
D --> E(Hunt)
```

```
A --> C
E --> A
A --> F(Incident Resoultion)
F --> C
```

Injects

Injects are most often tasks associated with system administration tasks. /example inject/

Each team is responsible for: - Resolving injects tasked to that team - Proper documentation - Communicating with other teams

Incidents

The entire team is responsible for maintaining fundamental services: - #### HTTP - #### HTTPS - #### Webmail-HTTP - #### SMTP - #### POP3 - #### DNS

Each team is responsible maintaining the services under their purview: - #### Firewall - Incoming and Outgoing Rules for all fundamental services - #### Windows - AD DNS - Webmail-HTTP - POP3 - #### Linux - HTTP/HTTPS - DNS - SMTP

Hardening

There should be no injects given within the first 15 minutes. Hardening each and every device under your purview is the first step that should be taken towards securing a system. Since hardening a system is never completely finished, break hardening up into an initial step and a recurrent process.

The initial step breaks down neatly into three smaller, consecutive steps:

1. new user passwords 2. configure admin accounts 3. access rights

These steps differ slightly per team.

- Network Admin

1. Change Default Credentials
2. Harden Admin Account
3. Define Firewall Rules

- Windows Team

1. Change Default Credentials
2. Create an Admin Account

3. Restrict Login Access

log successful and failed logins

```
auditpol.exe /set /category:"Logon/Logoff"  /success:enable /failure:enable | out-null
```

- Linux Team

1. change default user credentials

```
# change default password for default login
passwd
# open a root shell and change root password
sudo -i
passwd
usermod -l <newname> <oldname>
usermod -d ~/home/<newname> -m <newname>
# symlink $HOME
ln -s ~/home/<newname> ~/home/<oldname>
```

2. configure wheel and add an admin

*add the **wheel** group if it doesn't already exist!*

```
groupadd wheel
```

*Restrict the use of sudo to the wheel group by configuring **/etc/sudoers**. Use **visudo** and uncomment the following:*

```
# option A: faster
root ALL=(ALL) ALL
wheel ALL=(ALL) ALL

# option B: arguably more secure
#root ALL=(ALL) ALL
wheel ALL=(ALL) NOPASSWD: ALL
```

*Restrict use of su with pam. Uncomment or add the following line to **/etc/pam.d/su**:*

```
auth        requirement pam_wheel.so group=wheel
```

while root create an admin account:

```
useradd -mg wheel <admin>
passwd <admin>
exit
# login as admin and restrict root login and su to <admin>
sudo -i -u <admin>
sudo passwd -l root
sudo chown <admin>:wheel /bin/su
```

****Use sudo -i -u adminname when performing admin tasks!***

3. Restrict Login Access

/etc/pam.d/system-login

```
# Set a delay upon authentication failure
# Lock out a user after 3 repeated failed attempts
auth optional pam_faildelay.so delay=4000000
auth required pam_tally2.so deny=3 unlock_time=600 onerr=succeed file=/var/log/tallylog
```

```
# secure ssh files access mode

chmod 700 ~/.ssh
chmod 600 ~/.ssh/*
chown -R $USER ~/.ssh
```

/etc/security/limits.conf

```
#Limit processes run by users
* soft nproc 100
* hard nproc 200
```

Enumeration

Firewall

Windows

Linux

nmap

```
# basic usage
sudo nmap <args> <ip_address>

# scan multiple hosts by using a comma
sudo nmap 192.168.0.1,8.8.8.8,8.8.4.4

# scan whole subnet
sudo nmap 172.20.201.0/24

# aggressive scan (time consuming)
# detects OS and services

sudo nmap -A <hosts>
# nmap specific port

sudo nmap -p <port> <host>
# nmap range of ports
```

```
sudo nmap -p <startport-endport> <host>
# example:
sudo nmap -p 1-100 192.168.1.254

#nmap 100 most common ports
sudo nmap -F <host>

#Service detection
sudo nmap -sV <host>
```

Maintaining Services

systemctl

```
# to get service status:
systemctl status <service>  # you can omit <service> to list all as a tree
# to list running services or failed services:
systemctl | grep running
systemctl --failed
# to start, stop, restart a status:
systemctl start <status>
systemctl restart <status>
systemctl stop <status>
# to enable or disable a service:
systemctl enable <service>
systemctl disable <service>
```

journalctl

```
# show all messages since 20 minutes ago:
journalctl --since "20 min ago"
# follow new messages:
journalctl -f
# show all messages by a specific executable:
journalctl /usr/lib/systemd/systemd
# show all messages by a specific process:
journalctl _PID=1
# show all messages by a specific unit:
journalctl -u <service>
# show kernel ring buffer:
journalctl -k
```

Hunting

Windows Team

Linux Team

tcpdump

```
# if packet capturing is needed we can do so on any linux device, but we'll mostly do that f
tcpdump -lnn -i any port ssh and tcp-syn
```

ss

```
# print socket statistics as they are destroyed
# this would be great if there were time to configure tmux or screen
ss -E

# summary of connections mainly for checking udp and tcp numbers
ss -s

# discover which user opened ssh
ss -lp | grep ssh

# show processes using sockets, this may reveal an uid on the rightmost column
# if so, follow up with the second command:
ss -ltpe
getent passwd | grep <uid>

# display all established SMTP connections
ss -o state established '( dport = :smtp or sport = :smtp )'

# display all established HTTP connections
ss -o state established '( dport = :http or sport = :http )'

# display all estalblished connections to the mySQL server
ss dst 172.20.240.20:3306
```

ps

```
# display processes by user
ps -fU <user>
ps -fu <uid>

# all processes running as root
ps -u 0 -u root

# detailed list of processes on tty1
ps -e --forest -ft tty1
```

```
# detailed list
ps -p 999 -e --forest -o pid,ppid,fgroup,ni,lstart,etime

# monitor processes
watch -n 1 'ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head'
```

kill kill -9

File Permissions

Every file and folder has an access mode that describes *who* can do *what* *who = user=u,group=g, or other=o what = read=r,write=w,execute=x*

	User	Group	Other
r	4	4	4
w	2	2	2
x	1	1	1

- 0777 means u=rwx g=rwx o=rwx ; everyone has read write execute permissions
- 0700 means u=rwx g=— o=—
- 0000 means lock the door and throw away the key

chmod

```
# change access mode of a folder or file
# general usage:
chmod [OPTION] MODE[,MODE] FILE

chmod 700
chmod 077 /boot /etc/{iptables,artptables}
chmod -R
```

chown

```
# chown differs from chmod in that it changes only the user and the not the access mode
# change ownership of a file to <user>:<group>
chown <user>:<group> /path/to/file
# change ownership to admin of a folder and all subfolders
chown -hR admin /directory
# exchange ownership of all files from <badguy> to <goodguy>
chown -R --from=<badguy> <gooduy> /
```

find

```
# Note: /path/to/file refers to any path (. ./ /home /etc ..)
# print files owned by a user
find /path/to/file -user <user>
```

```
# print .ext files owned by a user
find /path/to/file -user <user> -name "*.ext"
# you can do the same but by group
find /path/to/file -group <group> -name "*.ext"
# print all shell files owned by usera and userb with ls formatting
find /path/to/file -name "*.sh" -user usera userb -ls
# you can also use logical operators
# -o -or, ! -not, -a -and (^^implicit without operators)
find /path/to/file -name "*.sh" -user usera -o userb -ls
# delete all files owned by a user
find /path/to/file -user <badguy> -delete
```

ufw

```
# genral usage
sudo ufw default <deny/allow> <incoming/outgoing> <port/protocol>

# disable or enable ufw like this
sudo ufw <enable/disable>

# show rules
sudo ufw status
sudo ufw status numbered
sudo ufw status verbose

# open ports like this
sudo ufw allow <port/protocol>

# allow ssh over tcp or udp
sudo ufw allow 22
sudo ufw allow 22/tcp
sudo ufw allow 22/udp

# close ports like this
sudo ufw deny <port/protocol>

# deny ssh
sudo ufw deny 22/tcp

# you can also allow or deny services in /etc/services
sudo ufw <allow/deny> <service name>
# example Usage:
sudo ufw allow ssh

# you can delete existing rules like this
sudo ufw delete allow 22/tcp
sudo ufw delete deny ssh
```

```
# you can delete existing rules by number
sudu ufw delete <n>
```

iptables

```
# open incoming traffic from port 80
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT

# accept incoming traffic from port 22 from <address> over tcp
iptables -A INPUT -p tcp -s <address> -m tcp --dport 22 -j ACCEPT

# deny outgoing traffic port 22 from <address>
iptabled -A OUTPUT -p tco -s <addrss> -m tcp --dport 22 -j DENY

# flush all rules
iptables -F
```

Maintaining Services

systemctl

```
# to get service status:
systemctl status <service>  # you can omit <service> to list all as a tree
# to list running services or failed services:
systemctl | grep running
systemctl --failed
# to start, stop, restart a status:
systemctl start <status>
systemctl restart <status>
systemctl stop <status>
# to enable or disable a service:
systemctl enable <service>
systemctl disable <service>
```

journalctl

```
# show all messages since 20 minutes ago:
journalctl --since "20 min ago"
# follow new messages:
journalctl -f
# show all messages by a specific executable:
journalctl /usr/lib/systemd/systemd
# show all messages by a specific process:
journalctl _PID=1
# show all messages by a specific unit:
journalctl -u <service>
```

```
# show kernel ring buffer:
journalctl -k
```