

[UNIX] Vulnerabilities Found in Sconly

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-08/0067.html>

From: support@securiteam.com

Date: 08/21/02

From: support@securiteam.com

To: list@securiteam.com

Date: Wed, 21 Aug 2002 10:55:45 +0200 (CEST)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

Vulnerabilities Found in Sconly

SUMMARY

<<http://www.sublimation.org/sconly/>> Sconly is an alternative 'shell' (of sorts) for system administrators who would like to provide access to remote users to both read and write local files without providing any remote execution privileges. Functionally, it is best described as a wrapper to the "tried and true" SSH suite of applications. A vulnerability in the product allows circumvention of access limitations through user's SSHd environment files.

DETAILS

Sconly makes no effort to verify the path to the scp or sftp-server executables before it executes them, and uses system() to do so.

If the server administrator makes no effort to restrict access to the user's .ssh directory, the user can upload a file with a custom environment to \$HOME/.ssh/environment. Subsequently a user can upload a script or program to run arbitrary commands so that they change their login shell, or what have you.

For example, the user could scp the following to \$HOME/.ssh/environment:

Securiteam: [UNIX] Vulnerabilities Found in Scponly

```
# ssh environment
PATH=/home/myhomedir/:/usr/bin:/bin
#end
```

Subsequently, the user could upload the following file to their home directory, and call it scp:

```
#!/bin/sh

echo "I'm a bad boy" > /tmp/exploit
/usr/bin/scp $@

# end
```

When they next scp a file:

```
[root@restricted /tmp]
# ls -l
total 24
-rw-r--r-- 1 bonehead bonehead 14 Aug 19 22:46 exploit
[root@restricted /tmp]
# cat exploit
I'm a bad boy
```

Provided they are careful about output of their command, with the above script, the file still is copied and anyone watching over their shoulder is none the wiser. Obviously, this could be replaced with any arbitrary command. This provides the user with a means of running arbitrary commands by simply uploading a file. Another neat trick is:

```
echo "mypassword" | chsh -s /bin/bash
```

Now the user can log in with SSH, assuming chsh allows users to change their own shells.

Additionally, some versions of the OpenSSH sshd(8) man page claim that at start-up, SSHd will execute commands in \$HOME/.ssh/rc using /bin/sh, rather than with the user's shell as listed in /etc/passwd. The man page on Derek's system says this, even though in practice the version of SSHd Derek has installed actually does use the user's shell. However, if this is **NOT** the case, the user could execute arbitrary commands by uploading a file to \$HOME/.ssh/rc.

Finally, though effort is made to remove shell meta-characters from the input, scponly uses system() to execute commands. Therefore, wildcards are possible. In some environments, it may be possible to exploit this situation.

Solution:

There are several possible ways to "fix" these problems, some perhaps better than others.

1. Limit the user's ability to affect their environment

The first is more a workaround than a solution, but should be effective. This is the route the author has chosen to go, for the moment, while he works on other fixes. The ability of the user to circumvent sconly is dependent upon their ability to manipulate their environment, by uploading files to specific locations in their .ssh directory.

The system administrator can prevent this by making the user's home directory non-writable to the user. In order to provide file upload to the user, a user-writable directory must be provided for that purpose.

Some may feel this is too restrictive; some may feel that creating .ssh/ and making it non-writeable to the user is sufficient. It is not. If the user has write access to their home directory, the user can log in via sftp, and simply remove the .ssh directory if it is empty, or rename it if it is not, regardless of who owns it or what its permissions are. The only way to prevent them from doing this is to make their home directory non-writable.

This will prevent the user from being able to modify their environment files, preventing the exploit.

Depending on how the call to system() can be exploited, if it can be at all, this may or may not solve that problem.

The author's update involves documenting the problem, and updating the installation to include some chown commands. The author does intend to remove the call to system() in the immediate future; but with no threat of a known exploit which was not fixable in doing the above, was not concerned that Derek wait to release this vulnerability before having the opportunity to do so.

2. Fix the code

These problems can be eliminated by forcing the use of the "correct" path for the scp or sftp-server binaries on the restricted host, and by using execv() to execute the programs. The correct paths can be either provided by configuration file, or compiled in. Eliminating the reliance on the user's environment prevents them from being able to make modifications that affect the program.

Incidentally, Derek became aware of the vulnerabilities in sconly after having written his own program, called rssh, to do essentially the same thing. Derek took this approach. Unfortunately, if SSHd on the system in question does in fact use /bin/sh to execute programs in \$HOME/.ssh/rc, there's not much you can do programmatically, and the workaround in part 1 must be used. Derek note this in the manpage for rssh. The limitation is that by the time sconly (or rssh) runs, SSHd has already allowed the user to execute commands.

Securiteam: [UNIX] Vulnerabilities Found in Scponly

For the interested, rssh can be downloaded here:

<<http://www.pizzashack.org/rssh/>> <http://www.pizzashack.org/rssh/>

3. "Fix" SSHd

While not technically broken, it surprises Derek that there is no option in OpenSSH's SSHd to ignore user environment variables, especially since they do have options to ignore other user files (specifically .rhosts and ssh/known_hosts). Used in conjunction with such an option, all of the problems above would be eliminated. For this sort of functionality, the combination of 2 and 3 is probably the ideal solution.

ADDITIONAL INFORMATION

The information has been provided by <mailto:ddm@pizzashack.org> Derek D. Martin.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- **Previous message:** support@securiteam.com: "[UNIX] Another Buffer Overflow Found in PostgreSQL (repeat function)"
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)