

[NEWS] ClamAV Multiple Vulnerabilities (Temporary Files, UUEncode)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2008-01/msg00001.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 2 Jan 2008 10:35:31 +0200
-

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

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.
<http://www.securiteam.com/maillinglist.html>

ClamAV Multiple Vulnerabilities (Temporary Files, UUEncode)

SUMMARY

Multiple vulnerabilities have been discovered in ClamAV, these vulnerabilities allow attackers to exploit a race condition or a insecure handling of files to overwrite arbitrary files, or use UUEncoding to prevent detection by the antivirus engine.

DETAILS

Vulnerable Systems:

- * ClamAV version 0.92

Race Condition Vulnerability Details

\$SOURCE/libclamav/others.c (line 488):

```
int cli_gentempfd(const char *dir, char **name, int *fd)
{
```

```
    *name = cli_gentemp(dir);
    if(!*name)
    return CL_EMEM;
```

[NEWS] ClamAV Multiple Vulnerabilities (Temporary Files, UUEncode)

```
*fd = open(*name, O_RDWR|O_CREAT|O_TRUNC|O_BINARY, S_IRWXU);
if(*fd == -1) {
cli_errmsg("cli_gentempfd: Can't create temporary file %s: %s\n",
*name, strerror(errno));
free(*name);
return CL_EIO;
}

return CL_SUCCESS;
}
```

This function, `cli_gentempfd`, uses a custom function to generate a (more or less) unique file name which is then opened, and the file descriptor is returned via an output parameter.

The problem with this code is that a race condition exists: if the attacker is able to guess the generated file name, he/she is able to create such a named file between the call of `cli_gentemp()` and `open()`, making it possible to overwrite arbitrary files to which the user that runs ClamAV has write access with temporary data. A solution to fix this problem is to use the `O_EXCL` option for `open()`. This option prevents that the file will be opened if it already exists.

So, how does the file name generation happen? First, `cli_gentemp()` determines the temporary directory. Users of the `cli_gentemp()` function can specify their own custom temporary directory. If none is specified, then the content of the `TMPDIR` environment variable is used. If the environment variable is unset, then `P_tmpdir` resp. `"/tmp"` are used. The generated format of the file name is `$TMPDIR/clamav-$HASH`, where `$HASH` is generated from a fixed 16 byte "salt" and 32 (more or less) random bytes.

The salt is defined in the following way:

```
static unsigned char name_salt[16] = { 16, 38, 97, 12, 8, 4, 72, 196, 217,
144, 33, 124, 18, 11, 17, 253 };
```

The random bytes are generated with an internal function `cli_rndnum()` which looks like this:

```
unsigned int cli_rndnum(unsigned int max)
{
struct timeval tv;

gettimeofday(&tv, (struct timezone *) 0);
srand(tv.tv_usec+clock());

return rand() % max;
}
```

As you can see, every time `cli_rndnum()` is called, the random number generator is reinitialized with the microsecond component of the current time and an "approximation of the processor time used by the program"

using the clock() function. This takes away a lot of randomness from the value returned by cli_rndnum(): as seed, more or less public information which should be relatively easy to be guessed by the attacker is used, making it possible to guess the value returned by rand(). Also, since the random number generator is reseeded every time cli_rndnum() is called, every returned value is directly computed from the seed.

In addition, cli_rndnum() uses the modulo operator to "cut off" the random number at a maximum value, which is discouraged by virtually every documentation of the rand() function. The publication "Numerical Recipes in C: The Art of Scientific Computing"[0] says about the use of rand():

```
"If you want to generate a random integer between 1 and 10, you should  
always do it by using high-order bits, as in  
j=1+(int) (10.0*rand()/(RAND_MAX+1.0));
```

```
and never by anything resembling  
j=1+(rand() % 10);
```

(which uses lower-order bits)."

The function cli_gentempfd() is used throughout the whole ClamAV source code in numerous places, which means that all these places are affected by the race conditions. Ironically, the code also uses cli_gentemp() in several places to generate a random file name and then passes the file name to call to open() with the O_EXCL option enabled.

The race condition was introduced to the ClamAV source code on August 31st, 2007, in SVN revision 3196. The first release that contains the bug was 0.92. Since then, the code has remained in the trunk of the SVN repository.

Base64 UUEncoded Files Scanner Bypassing Details

ClamAV contains functionality to unpack and scan different types of files, such as archive files. Beside others, UUEncoded files are supported, too.

But ClamAV a variation of the UUEncoded file format: uuencode implementations, such as the one supplied by the GNU sharutils package, support the use of BASE64 instead of the uuencode-proprietary encoding, making it possible to create uuencode files slightly smaller. These files start with "begin-base64" instead of the usual "begin " of UUEncoded files.

When ClamAV tries to determine a file's type through its magic number at the beginning, it correctly recognizes files that start with "begin ", but fail to recognize Base64-UUEncoded files, since it's not in the list of checked magic numbers.

This makes it possible for attackers to bypass scanning of archive files by packing e.g. malware into a Base64-UUEncoded file, which the user then opens.

[NEWS] ClamAV Multiple Vulnerabilities (Temporary Files, UUEncode)

As history has shown, social-engineering-based attacks with unknown file attachments have been successful a lot of times. This fact makes such an attack sound more realistic, especially since many users blindly trust their virus scanner. In cases where ClamAV is the only virus scanner in the mail delivery process, and then fails to scan Base64-UUEncoded malware, this would be even more serious, since naive users may open unknown files without suspicion "since the virus scanner has checked them already and found nothing".

Insecure File Handling in Sigtool Details

Sigtool as shipped in a ClamAV installation is vulnerable to insecure file handling in some cases. If Sigtool is used with the 'utf16-decode' it fails to check if the <filename>.ascii file already exists and thus allowing an attacker with write permissions to the directory in which <filename> is located (for example if Sigtool is used in /tmp/) to overwrite arbitrary files belonging to the user via a symlink attack.

sigtool.c:

```
156 fname = opt_arg(opt, "utf16-decode");
157 if((fd1 = open(fname, O_RDONLY)) == -1) {
158 mprintf("!utf16decode: Can't open file %s\n", fname);
159 return -1;
160 }
161
162 newname = malloc(strlen(fname) + 7);
163 sprintf(newname, "%s.ascii", fname);
164
165 if((fd2 = open(newname, O_WRONLY|O_CREAT|O_TRUNC, S_IRWXU)) < 0) {
166 mprintf("!utf16decode: Can't create file %s\n", newname);
```

open is missing O_EXCL and thanks to the usage of O_TRUNC the file will be truncated to size zero.

References

[0]: Numerical Recipes in C: The Art of Scientific Computing (William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling; New York: Cambridge University Press, 1992 (2nd ed., p. 277))

ADDITIONAL INFORMATION

The information has been provided by <<mailto:lolek1337@xxxxxxxxxxxxxxxx>>
Lolek of TK53.

=====

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

[NEWS] ClamAV Multiple Vulnerabilities (Temporary Files, UUEncode)

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

list-unsubscribe@xxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====
=====

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.