

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-02/msg00032.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 13 Feb 2007 12:19:38 +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>

Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion
DoS (Problem 3000)

SUMMARY

A vulnerability in the way Microsoft's Visual C++ handles large `time_t` values, allows attackers to overflow the variable, which in turn will cause the value stored inside the `time_t` variable to be invalid/corrupt, anyone using this value will get an assertion which in turn causes a denial of service vulnerability in the compiled program.

DETAILS

Introduction:

Since Microsoft Visual Studio 5.0, Visual C++ compiler defaults `time_t` type to 64 bit integer and time functions to their 64-bit variants.

Vulnerability:

64-bit versions of time functions:

`localtime()`

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

localtime_s()

gmtime()

gmtime_s()

ctime()

ctime_s()

wctime()

wctime_s()

fstat()

and may be others

incorrectly behave for a time_t argument larger than or equal to `__MAX__TIME64_T` (representing January, 1 3000 00:00:00). According to MSDN documentation, time functions must indicate error by returning NULL pointer or `EINVAL` (depending on function class) and must not invoke any invalid parameter handler. Instead, time function calls invalid parameter `assert()`-like macro, terminating calling application and creating Denial of Service condition for calling application.

An example is within `localtime_s` function (`loctim64.c`):

```
/*  
* Check for illegal __time64_t value  
*/  
_VALIDATE_RETURN_ERRCODE_NOEXC( (*ptime >= 0), EINVAL);  
_VALIDATE_RETURN_ERRCODE( (*ptime <= __MAX__TIME64_T), EINVAL);
```

Last string initiates assertion, it's invalid

`_VALIDATE_RETURN_ERRCODE_NOEXC` must be used for both negative and oversized value. Valid code is:

```
/*  
* Check for illegal __time64_t value  
*/  
_VALIDATE_RETURN_ERRCODE_NOEXC( (*ptime >= 0), EINVAL);  
_VALIDATE_RETURN_ERRCODE_NOEXC( (*ptime <= __MAX__TIME64_T),  
EINVAL);
```

Both static and dynamic (`MSVCR80.DLL`) versions of C library are vulnerable.

Who is vulnerable?

Any application compiled with Microsoft Visual C++ 8.0 compiler with either static or dynamic libraries is vulnerable, if it uses one of named functions with user-controlled data.

Possible attack vectors:

1. Network protocols and applications where `time_t` value is used and/or transmitted as 8-octets (64 bit) in seconds or milliseconds and can be

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

behind January, 1, 3000. Example: different SQL databases.

2. Windows applications where `time_t` is result of conversion from `FILETIME` or `SYSTEMTIME` structures. E. g. `GetFileTime/SetFileTime` functions can be used to get/set NTFS file time to values behind January, 1, 3000. You can try to exploit different applications by using this very simple trick. This is also true for Java and JavaScript timestamps.

3. Application where `date_t` is calculated as a result from user input + some offset (e.g. timezone conversions for date December, 29 2999 23:01 GMT-01:00). An example: e-mail messages, HTTP requests, etc.

Example:

```
/*
```

```
D:\>cl localtime_s.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 14.00.50727.42 for
80x86
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
localtime_s.c
Microsoft (R) Incremental Linker Version 8.00.50727.42
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
/out:localtime_s.exe
localtime_s.obj
```

```
D:\>localtime_s.exe
```

(Dr.Watson comes, expected result: "Invalid value")

```
*/
```

```
#include <time.h>
#include <stdio.h>
#include <error.h>
```

```
int main(){
struct tm tm;
time_t t = 0x3a3a3a3a3a3a3a3a;
```

```
if(localtime_s(&tm, &t) != 0) {
printf("Invalid value\n");
}
else {
printf("OK\n");
}
return 0;
}
```

Workarounds:

Developer can use one if this workarounds:

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

1. Define `_USE_32BIT_TIME_T` to use 32-bit functions (not available on 64-bit platforms).
2. Explicitly check 'time' argument of named functions to be below `_MAX__TIME64_T`. It should be noted, that this workaround is not reliable, because it doesn't covers the vector where `time_t` is calculated as a result of time arithmetics.

Exploitation:

Test application to set file date to 27.09.14896 3touch.c is available from <http://SecurityVulns.com/news/MICROSOFT/Time/Assert.html>
<http://SecurityVulns.com/news/MICROSOFT/Time/Assert.html>.

Application compiled with MSVC 8.0, e.g. MSDN sample `fstat.c`, crashes on attempt to `fstat()` this file.

It may also be used to get interesting results with "dir" command (shows "Invalid argument") if `FILETIME` is changed to `0x7FFFFFF00`, but it seems to be different issue.

Vendor:

23.08.2006 – Initial vendor notification through `secure@xxxxxxxxxxxxxx`
25.08.2006 – Second vendor notification
25.08.2006 – Initial vendor reply
30.08.2006 – Vendor asks for additional details
31.08.2006 – Additional details with example of crashing application are sent to vendor
12.09.2006 – Additional details are sent again because of no response
11.10.2006 – Vendor response:

"We believe this is not a security vulnerability but in fact a deliberate security feature to mitigate problems with invalid data propagating through the system".

ADDITIONAL INFORMATION

The information has been provided by <mailto:3APA3A@xxxxxxxxxxxxxxxxxx>
3APA3A.
The original article can be found at:
<http://securityvulns.com/advisories/year3000.asp>
<http://securityvulns.com/advisories/year3000.asp>

=====

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@xxxxxxxxxxxxxx`

[NT] Microsoft Visual C++ 8.0 Standard Library Time Functions Invalid Assertion DoS (Problem 3000)

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.