

Visual Studio 6.0 Buffer Overflow Vulnerability

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2006-03/msg00097.html>

- *From:* kozan@xxxxxxxxxxxxxxxxxxxxx
 - *Date:* 4 Mar 2006 00:46:40 -0000
-

Visual Studio 6.0 Buffer Overflow Vulnerability

Bug Discovered by Kozan
Credits to ATmaCA
Web: www.spyinstructors.com
Mail: kozan@xxxxxxxxxxxxxxxxxxxxx

Affected Vendor:

Microsoft (www.microsoft.com)

Affected Products:

Microsoft Visual Studio 6.0 (with latest Service Pack 6)
Microsoft Development Environment 6.0 (SP6) (Microsoft Visual InterDev 6.0)

Vulnerability Details:

A Buffer Overflow Vulnerability is exists for the following file formats of affected product.

Visual Studio Database Project File (.dbp)
Visual Studio Solution (.sln)

The vulnerability is caused due to a boundary error within the handling of a ".dbp" file (.sln files are also affected) that contains an overly long string in the "DataProject" field. This can be exploited to cause a stack-based buffer overflow and allows arbitrary code execution when a malicious ".dbp" file is opened. A specially crafted project file can overwrite a stack based buffer allowing for fully EIP register control and code execution and compromise user's system.

An example .dbp file:

```
# Microsoft Developer Studio Project File - Database Project
```

Visual Studio 6.0 Buffer Overflow Vulnerability

```
Begin DataProject = "ProjectName"  
End
```

Carriage return and line feed (0x0d and 0x0a) characters and some others (0x00 ...) can not be used in project name variable.

An example .dbp file which overwrites EIP register:

```
# Microsoft Developer Studio Project File – Database Project  
Begin DataProject =  
"Project1AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAA123456AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  
End
```

The length must be 384 bytes long. Otherwise other registers will be overwritten differently and exploitation method will be changed. So 384 bytes long length is the most suitable way.

In this example when file is opened:

XXXX (0x58585858) characters will overwrite EIP.
And 123456AAA... (3132333435364141... in hex) bytes will be on ESP.

So an attacker could create a malicious .dbp project file which includes a payload which on ESP and EIP should point to this shellcode with a loaded moduls jmp esp or call esp opcodes.

PoC:

The local path length of the dbp file changes the arrangement of malformed data. So, exploit has to re-align the data for total path length.

Copy the following file as c:\deneme\Project1.dbp

<http://www.spyinstructors.com/kozan/poc/vuln.dbp>