



Desktop Development » Miscellaneous » General (CPOL)

License: [The Code Project Open License](#)

C#, ASM, Windows (WinXP, Win2003, Vista, Win2008, Win 7, Win2008 R2), ASP.NET, WinForms, Architect, DBA, Dev, QA

NET Debugging: Dump all strings from a Managed Code Process running

By [RenePallyZ](#)

Advanced Debugging Processes

Version: **3** ([See All](#))
 Posted: **1 Oct 2009**
 Views: **90**
 Bookmarked: **0 times**
[Unedited contribution](#)

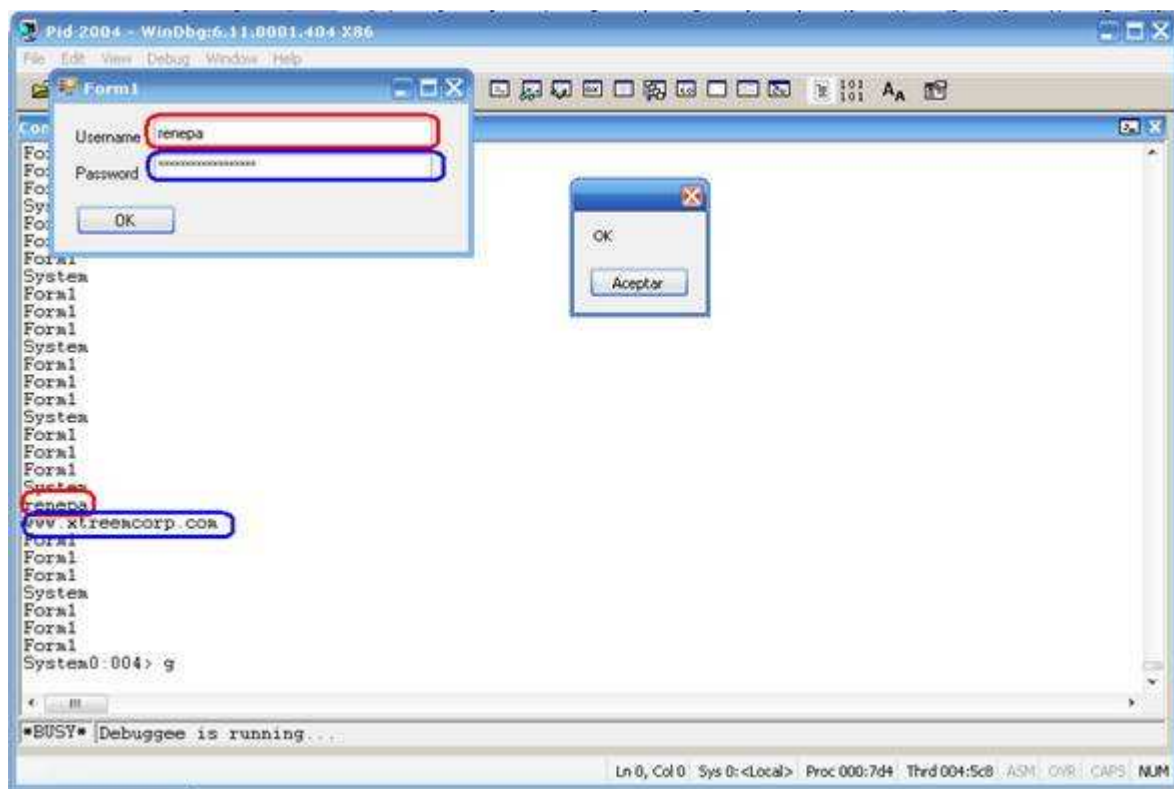
0 votes for this article

- [Download SampleAPP - 3.89 KB](#)

.NET Debugging: Dump all strings from a Managed Code Process running

Sometimes we would like to know which critical information is in our process, like bank accounts or credit card numbers it because we need to protect this information from malicious people, or just by simple exploration which internal strings are there. The task is very simple. In the picture we revealed the hidden text in the password. The procedure is applicable for every single .NET Running Process including IIS Worker Processes. If you need support please write me rpally@xtreemcorp.com

Final Output Picture:



System.String class is one of the most used classes in .NET World. I cannot imagine a simple or Enterprise Software that do not use this.

We will use Windows Debugger and a simple script to do this task.

1. Launch Windbg (The favourite tool that Microsoft Support uses) if you do not have this, please download from Microsoft Site
2. Copy the file %windir%\Microsoft.NET\Framework\v2.0.XXXXXX\SOS.DLL to %programfiles%\Debugging tools for windows\
3. Execute the sampleApp.exe attached to this mail (This is a dummy sample app)

4. Fill username and password e.g. username=renepa and password= www.xtreemcorp.com
5. Press OK button.
6. Return to Windbg and press F6 to Attach a file. ζ
7. In the process list choose SampleAPP.exe
8. In the Windbg put the command : .load sos it allows to load the extension SOS.DLL which is a .NET Debugging helper file. It contains a lot of new commands and facilities to debug .NET Apps.
9. SOS.DLL contains a very interesting command, !dumpheap, which allows to dump all classes which are in memory. It traverses the .NET Heap and dumps all the allocated objects. It also can be used to detect high memory usage. We will explain this in other articles.
 1. **!dumpheap -type System.String** This command allow us to dump all the memory addresses for a System.String classes. The output will be:

```

pid 2004 - WinDbg:6.11.0001.404 X86
File Edit View Debug Window Help
Command
0:004> !dumpheap -type System.String
Address      MT          Size
014d1198 793308ec   20
014d11c8 793308ec  120
014d1240 793308ec  152
014d13b8 793308ec   28
014d13d4 793308ec   32
014d13f4 793308ec   20
014d1408 793308ec   52
014d143c 793308ec   40
014d1464 793308ec   48
014d1494 793308ec   44
014d14c0 793308ec   40
014d14f0 793308ec   36
014d1514 793308ec   32
014d1534 793308ec   48
014d1564 793308ec   56
014d159c 793308ec   64
014d15dc 793308ec   36
014d1600 793308ec   44
014d162c 793308ec   64
014d166c 793308ec   68
0:004> |!dumpheap -type System.String |
Ln 0, Col 0 Sys 0: <Local> Proc 000:7d4 Thrd 004:ec: ASM OVR CAPS NUM

```

2. Please take a look at this output: Address (The pointer to the String class), MT=Method Table, which is the pointer to the list of Methods that System.String supports (We are not going to use MT at this time), and Size of this class
3. **!dumpheap -type System.String -short** This command will show just the memory addresses for the strings currently in memory.
4. **!do[memoryaddress]** This command will dump the object from memory. Lets explore the second string: e.g. : **!do 014d11c8**

```

pid 2004 - WinDbg:6.11.0001.404 X86
File Edit View Debug Window Help
Command
0:004> !do 014d11c8
Name: System.String
MethodTable: 793308ec
EEClass: 790ed64c
Size: 118(0x76) bytes
(C:\WINDOWS\i\assembly\GAC_32\mscorlib\2.0.0.0_b77a5c561934e089\mscorlib.dll)
String: C:\blog\dumpstrings\SampleAPP\SampleAPP\bin\Debug\
Fields:
MT      Field      Offset      Type      VT      Attr      Value Name
79332b38 4000096     4           System.Int32 1 instance 51 m_arrayLength
79332b38 4000097     8           System.Int32 1 instance 50 m_stringLength
793315cc 4000098     c           System.Char 1 instance 43 m_firstChar
793308ec 4000099    10           System.String 0 shared static Empty
>> Domain:Value 00165498:014d1198 <<
7933151c 400009a    14           System.Char[] 0 shared static WhitespaceChars
>> Domain:Value 00165498:014d1790 <<
0:004>
Ln 0, Col 0 Sys 0: <Local> Proc 000:7d4 Thrd 004:ec: ASM OVR CAPS NUM

```

5. The string is: **C:\blog\dumpstrings\SampleAPP\SampleAPP\bin\Debug** . But we are interested just in the chars itself not in the complete class. Where are located the characters? Looking a little the class we found that the m_first char is in the STRING MEMORY ADDRESS+c (Hexadecimal values). We can confirm this by exploring the memory address **014d11c8+c=014d11d4**

