Safe C API—Concise solution of buffer overflow

李建蒙

jimmyleeeeeee@gmail.com

Agenda

- Brief introduction of buffer overflow
- The difference between standard C API and Safe C API
- How does Safe C avoid buffer issue
- Cautions
- Summary

Buffer overflow

What is buffer overflow

- More data is put into a holding area than it can handle.

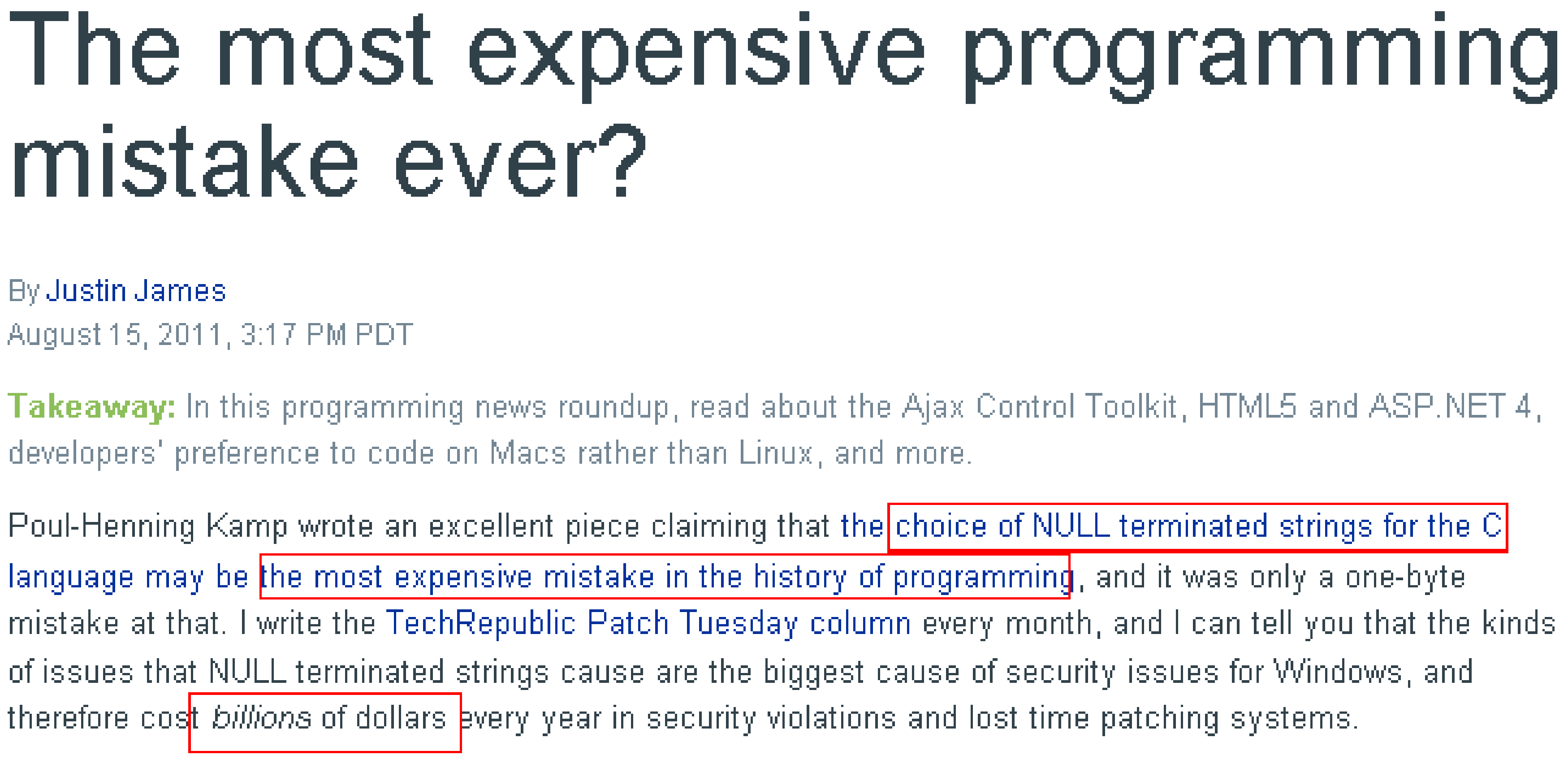
What’s the result of buffer overflow

- Programs can act in strange ways.
- Programs can fail completely.
- Programs can proceed without any noticeable difference in execution.

Notorious attack

Attack	Date	Damage
Morris Worm	1988-11	Over 6000 server crash Unix sendmail、Finger、rsh/rexec
Code Red worm	2001-7	IIS 4.0 and 5.0 allowing the worm to execute arbitrary code and infect the machine. It affected almost 1,500,000 system.
Slammer Worm	2003-1	Microsoft SQL Server 2000 a computer worm that caused a denial of service on some Internet hosts and dramatically slowed down general Internet traffic. Infect 359,000
Sun Solaris telnet daemon	2007-2	This may allow a remote attacker to trivially bypass the telnet and login authentication mechanisms.
Ubuntu Perl-Compatible Regular Expression (PCRE) library	2010-4	it could still be injected deliberated in malware to create backdoor entrances into a network

The cost of buffer error



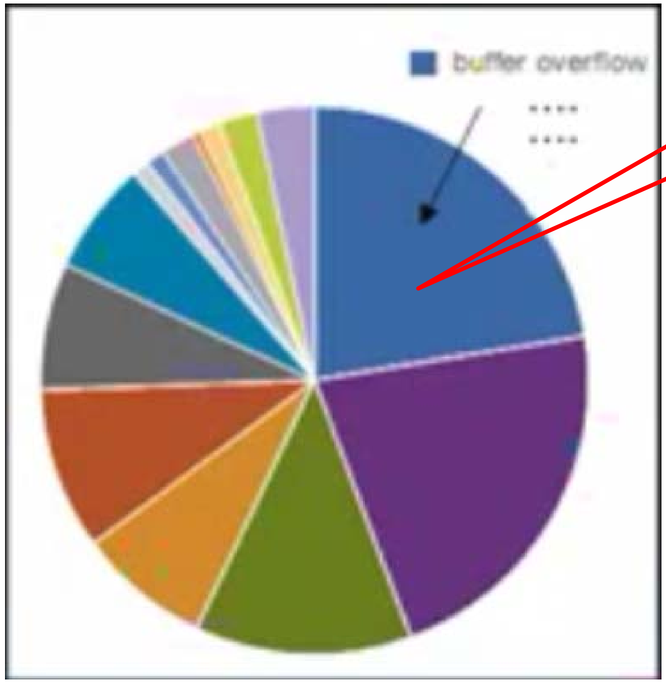
The most expensive programming mistake ever?

By Justin James
August 15, 2011, 3:17 PM PDT

Takeaway: In this programming news roundup, read about the Ajax Control Toolkit, HTML5 and ASP.NET 4, developers' preference to code on Macs rather than Linux, and more.

Poul-Henning Kamp wrote an excellent piece claiming that the choice of NULL terminated strings for the C language may be the most expensive mistake in the history of programming, and it was only a one-byte mistake at that. I write the TechRepublic Patch Tuesday column every month, and I can tell you that the kinds of issues that NULL terminated strings cause are the biggest cause of security issues for Windows, and therefore cost *billions* of dollars every year in security violations and lost time patching systems.

Buffer overflow bug



Buffer overflow

How to avoid it?

CPU/OS

- AMD Ehanced Virus Protection / Intel Excute Disable Bit (EDB)
- OS Data Execution Protection(NX)

Compiler

- MS: /GS /DYNAMICBASE /NXCOMPAT
- Linux: FORTIFY_SOURCE StackGuard StackShield ProPolice

Use different languages, like Java, C#

Write the right code

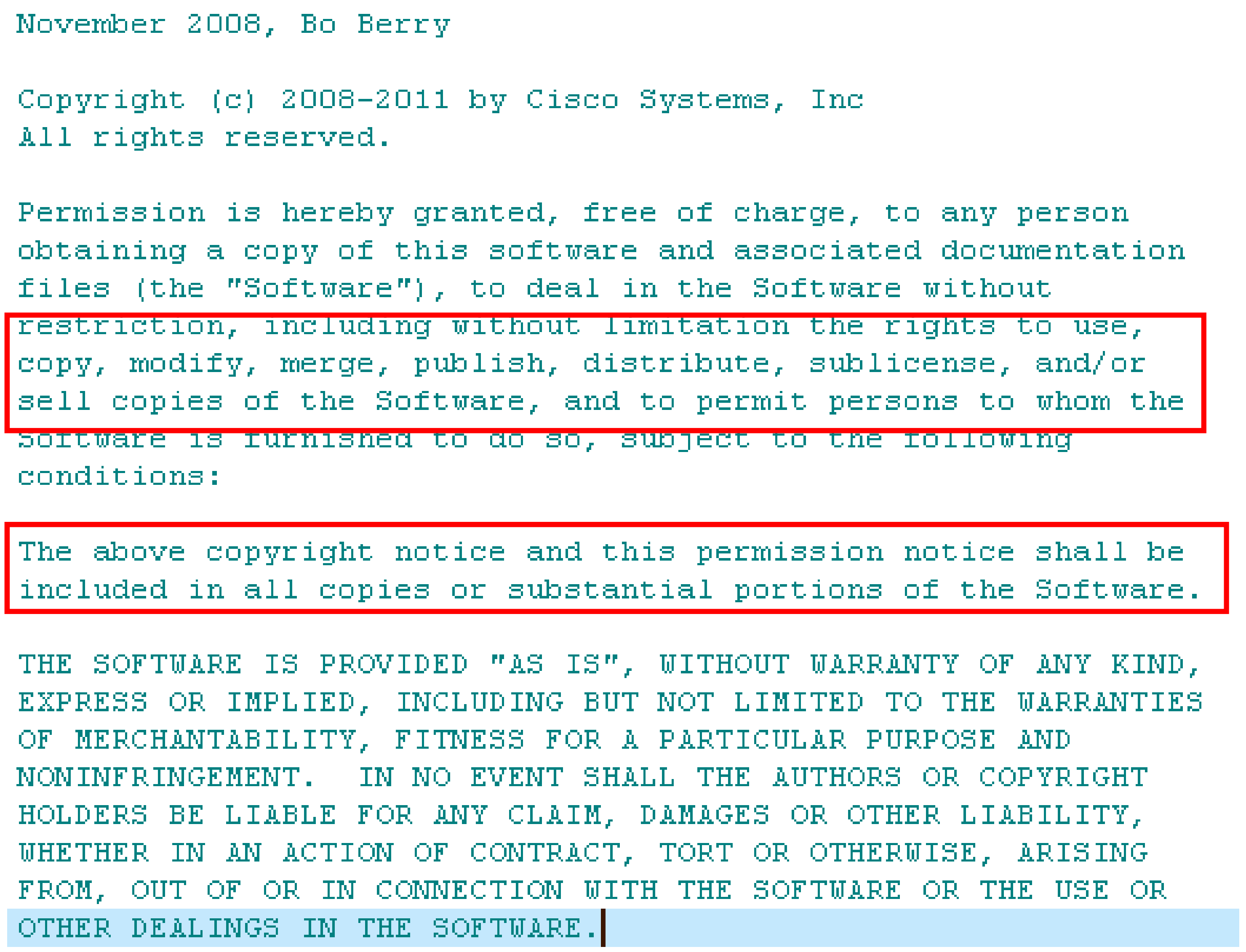
- Use safe library

  C++ STL

  C Safe C Library

  http://sourceforge.net/projects/safeclib

Safe C License



November 2008, Bo Berry

Copyright (c) 2008-2011 by Cisco Systems, Inc
All rights reserved.

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.

Overview the difference

Sample safe C replacement to traditional standard C lib function

C Standard	Safe C Standard
char *strcpy (dest, src)	errno_t strcpy_s (dest, dmax, src)

Error codes to indicate specific failure

_s postfix for all safe functions

Max destination buffer size to prevent overflow

The standard C API

Strong points

Convenient to use.

Performance is a little better than Safe C API

Weak points

No input validation, easy to cause buffer issue.

Some APIs have no return value to check whether there is an issue happened.

The Safe C API

Strong points

Input validation to avoid buffer issue, like overflow, un-terminated string etc.

Have return value to check whether there is an issue during calling.

Weak points

Performance is a little poorer than standard API.

Write more codes to check the return value.

Why Safe C?

Guard against overflowing a buffer

Do not unexpectedly truncate string

Do not produce un-terminated string

Return value to show whether there is error happened

Provide unified Runtime-constraint handler

string API List

Standard C API	Safe C API
strcpy	strcpy_s
strcat	strcat_s
strcmp	strcmp_s
stricmp	strcasecmp_s
strcmp	strcmp_s
strcspn	strcspn_s
strncat	strncat_s
strncpy	strncpy_s
strlen/strnlen	strnlen_s
strpbrk	strpbrk_s
strspn	strspn_s
strstr	strstr_s
strtok	strtok_s
strchr	strfirstchar_s
strrchr	strlastchar_s

How does string API to avoid buffer issue

strcpy_s

The difference to copy string

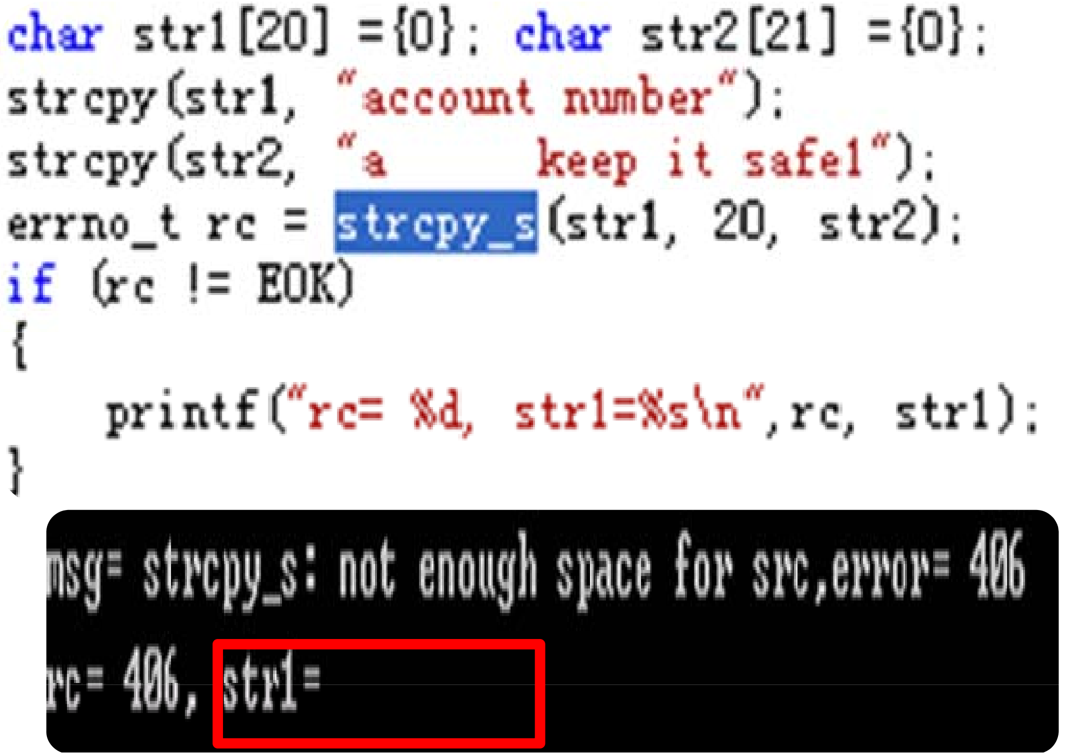
- Standard C

```
char str1[20] = {0};
char str2[20] ={”Just a test”};
strcpy(str1,   “a string“  );
```

- Safe C

```
errno_t rc =strcpy_s(str1, 20, str2);
if (  rc != EOK)   {/* copy failed */ }
else               {/* copy success */ }
```

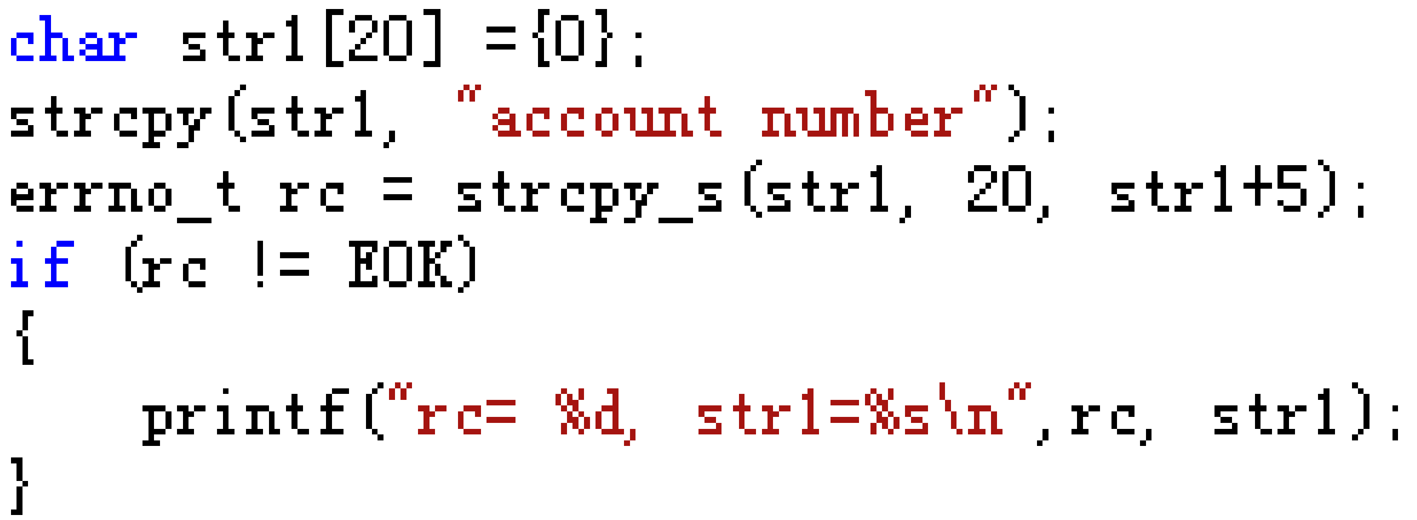
How does strcpy_s avoid buffer


```
char str1[20] ={0}; char str2[21] ={0};
strcpy(str1, "account number");
strcpy(str2, "a      keep it safe1");
errno_t rc = strcpy_s(str1, 20, str2);
if (rc != EOK)
{
    printf("rc= %d, str1=%s\n",rc, str1);
}
```

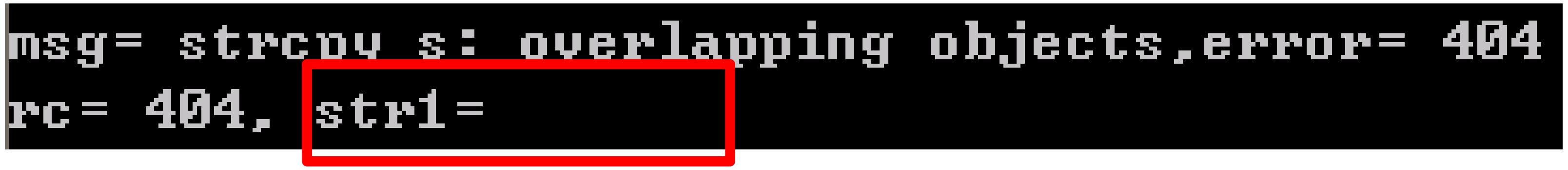
```
msg= strcpy_s: not enough space for src,error= 406
rc= 406, str1=
```

If Safe C check the buffer is not enough to contain the string(including the end char '\0'), it will empty the dest string.

How does strcpy_s avoid buffer issue --overlap



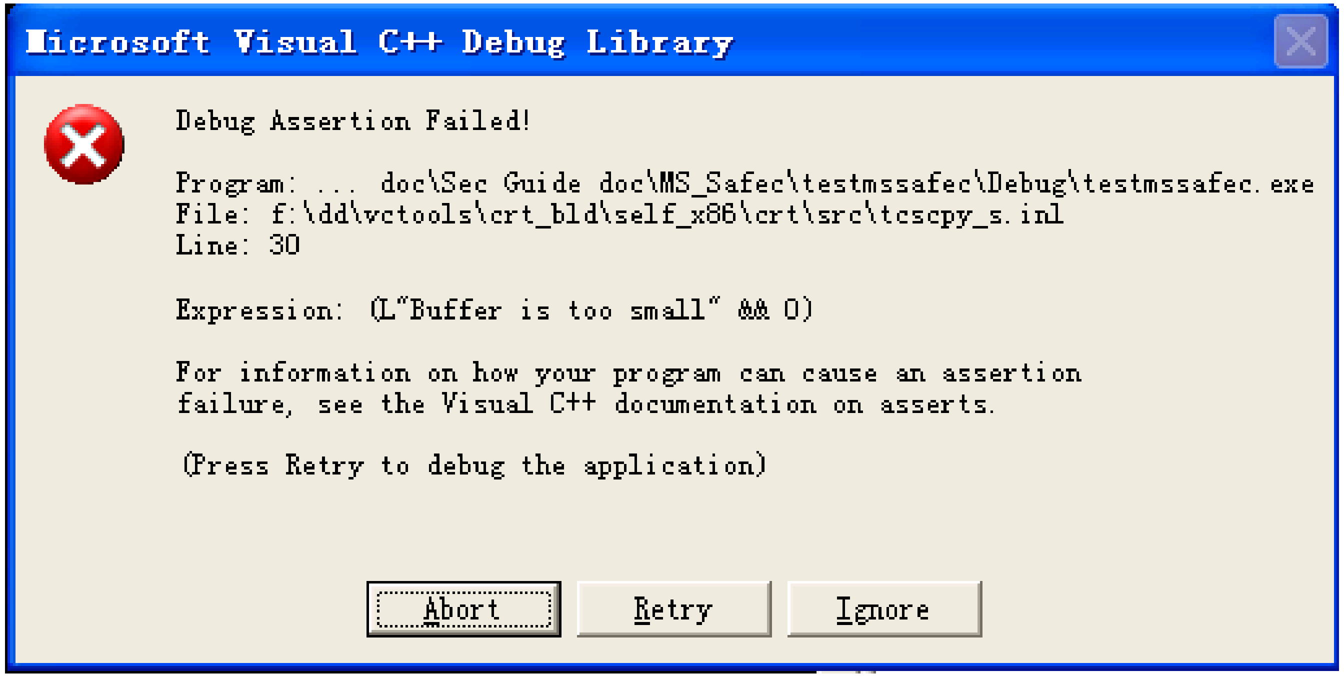
```
char str1[20] ={0};
strcpy(str1, "account number");
errno_t rc = strcpy_s(str1, 20, str1+5);
if (rc != EOK)
{
    printf("rc= %d, str1=%s\n",rc, str1);
}
```



```
msg= strcpy_s: overlapping objects,error= 404
rc= 404, str1=
```

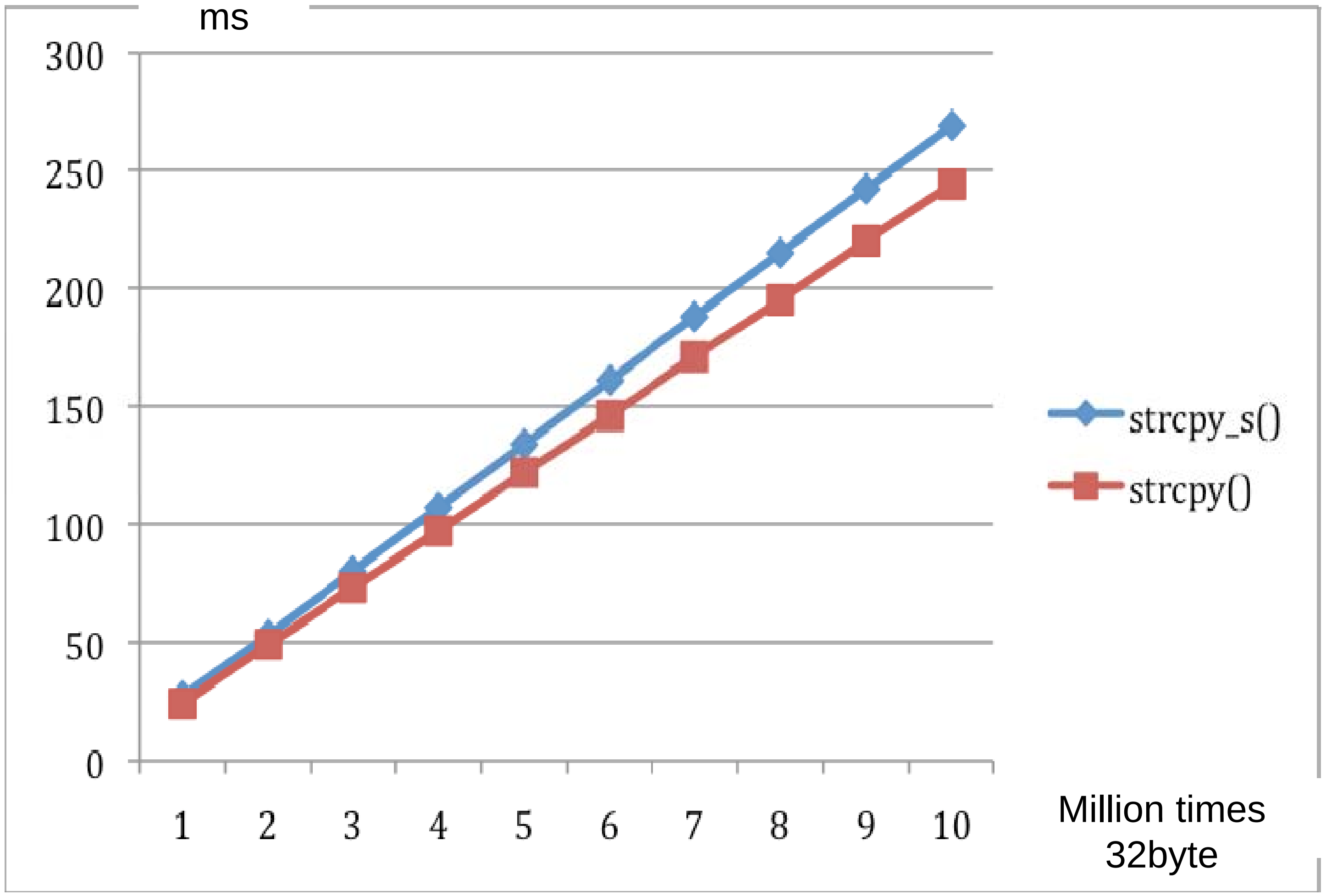
If Safe C check the buffer is overlapped, it will empty the dest string.

strcpy_s of MS


Microsoft Visual C++ Debug Library

```
Debug Assertion Failed!

Program: ... doc\Sec Guide doc\MS_Safec\testmssafec\Debug\testmssafec.exe
File: f:\dd\vctools\crt_bld\self_x86\crt\src\tcscpy_s.inl
Line: 30

Expression: (L"Buffer is too small" && 0)

For information on how your program can cause an assertion
failure, see the Visual C++ documentation on asserts.

(Press Retry to debug the application)
```

strcpy/strcpy_s Performance



How does string API avoid buffer issue

Which API will cause to set dest to empty?

- strcpy_s strncpy_s strcat_s strncat_s
- strcpyfld_s strcpyfldin_s strcpyfldout_s

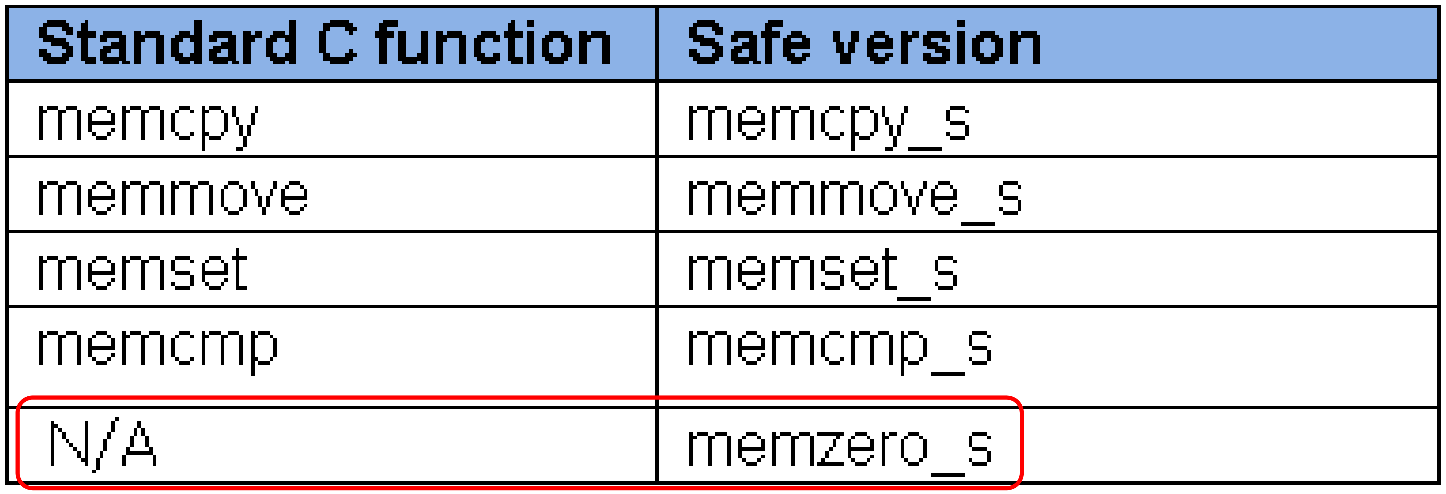
What kinds of error will set dest to empty?

ESOVRLP	Buffer overlap
ESUNTERM	unterminated string Like: dest[dmax-1] is not '\0'
ESNOSPC	not enough space

What is the default action?

- The default will only set the first byte to '\0'
- if want all bytes were set to '\0', please define SAFE_LIB_STR_NULL_SLACK.
- Or redefine the error handler.

Memory API list



Standard C function	Safe version
memcpy	memcpy_s
memmove	memmove_s
memset	memset_s
memcmp	memcmp_s
N/A	memzero_s

How does memory API avoid buffer issue

Which API will cause to set dest content to 0?

- memmove_s
- memcpy_s

What kinds of error will cause to set dest to 0?

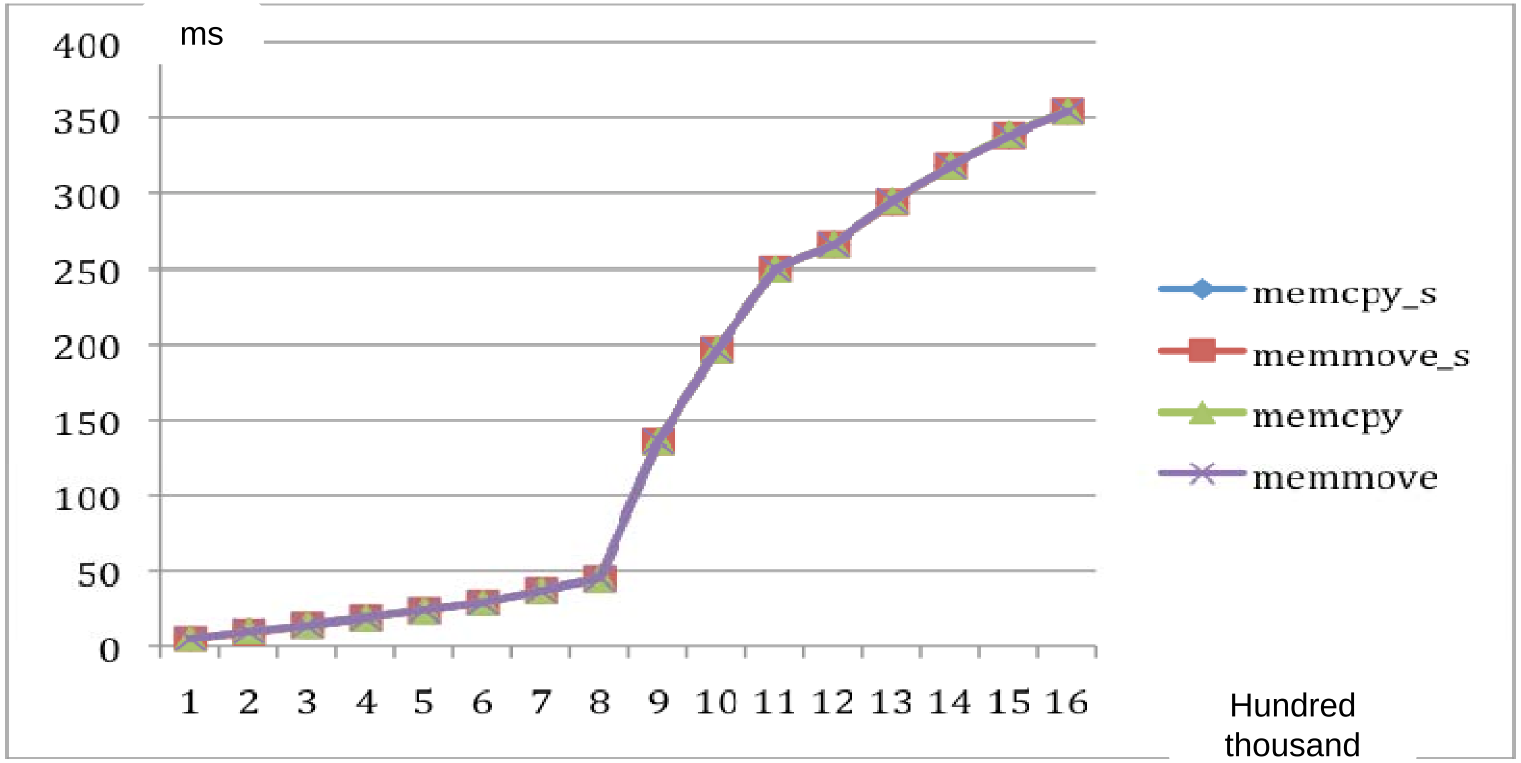
ESZEROL	smax is 0
ESNULLP	src is NULL
ESLEMAX	smax exceeds dmax
ESOVRLP	Memory overlap

What is the default action?

- the default will set all bytes to 0

Memory API- Performance

For Safe C memcpy_s and memmove_s are same. They call the same API.



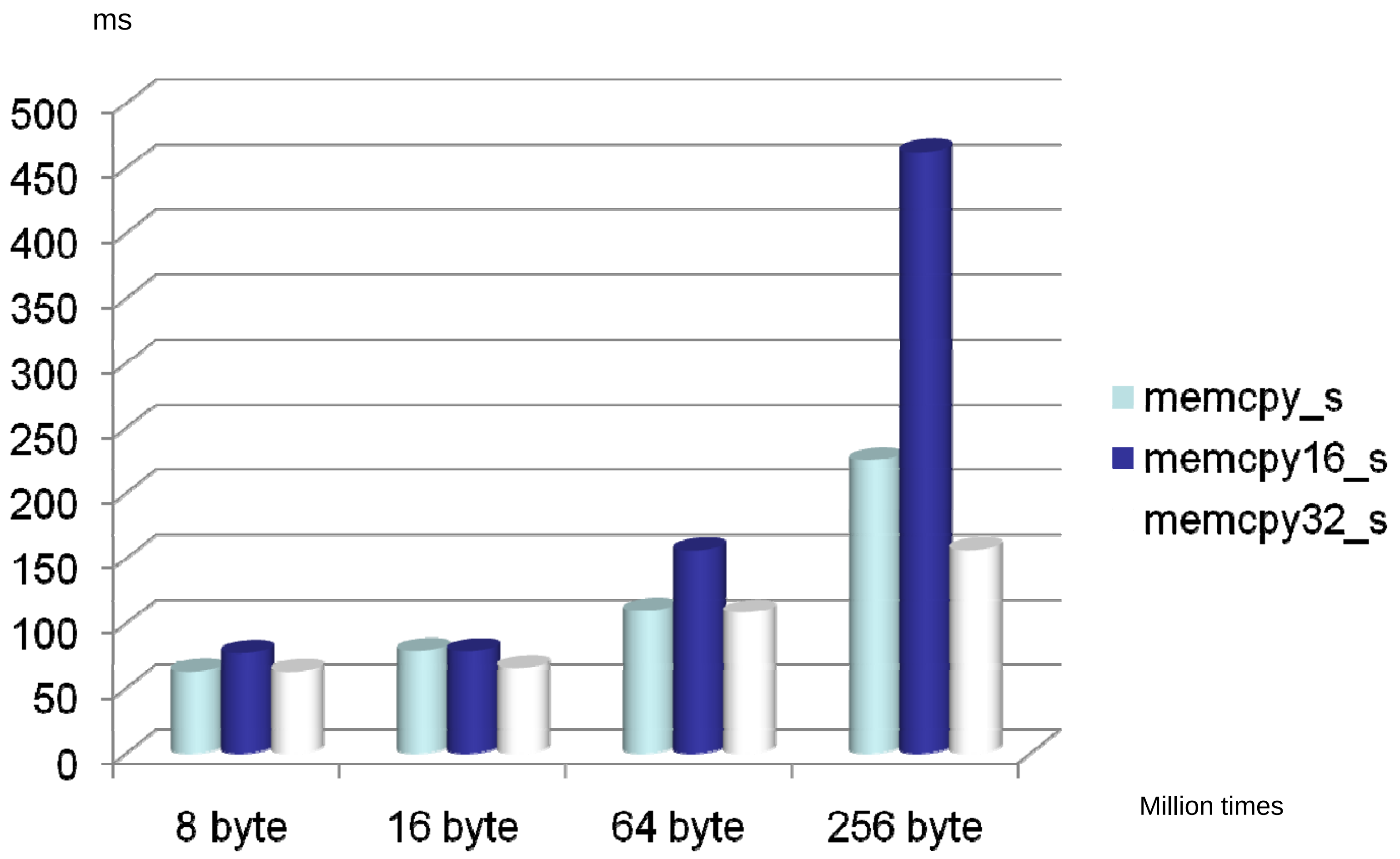
memcpy_s, memmove_s vs memcpy, memmove

Memory API

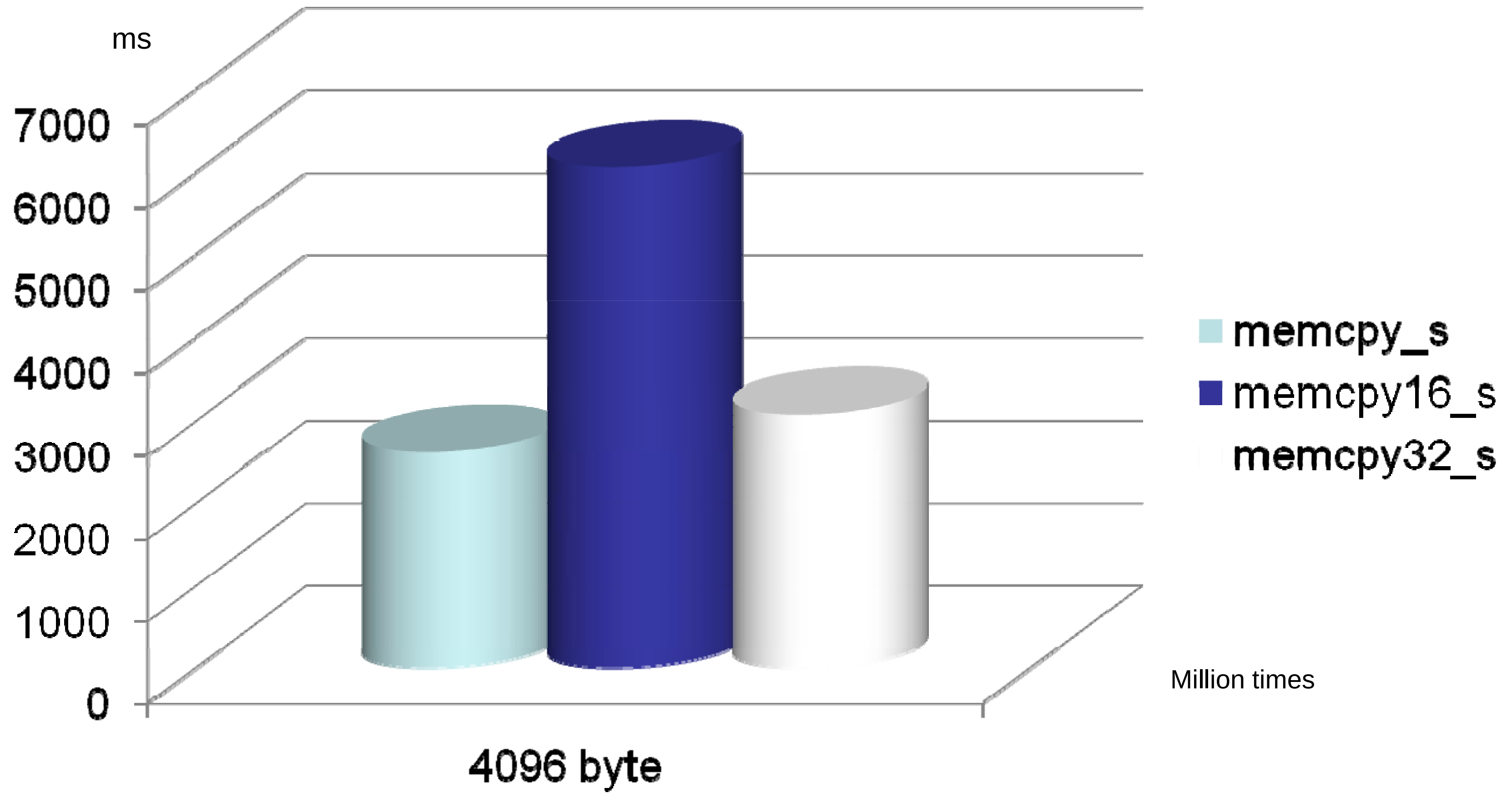
There are 3 APIs were provided by safe C for every standard API

Void *	Uint16 *	Uint32 *
memcpy_s	memcpy16_s	memcpy32_s
memmove_s	memmove16_s	memmove32_s
memset_s	memset16_s	memset32_s
memzero_s	memzero16_s	memzero32_s

Memory API- Performance test result I



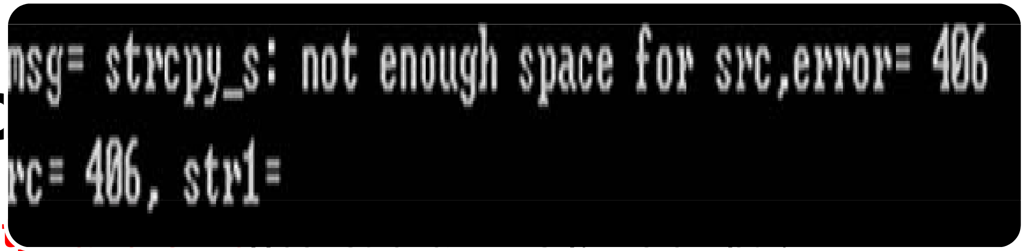
Memory API- Performance test result II



Error handler

The default handler

Simple error message to console



```
msg= strcpy_s: not enough space for src,error= 406
rc= 406, str1=
```

How to use your c

typedef void (***safe_lib_constraint**
errno_t error);

safe_lib_constraint_handler_t **safe_lib_set_constraint_handler**(

safe_lib_constraint_handler_t handler)

What kinds of platforms can use

Windows

MAC

Linux

Solaris

AIX

How to use it on Solaris?

The type in safe_types.h has conflict with inttypes.h

- int8_t int16_t int32_t uchar_t uint8_t uint16_t uint32_t ushort
- ulong ulonglong rsize_t

- **safe_types.h**
- #ifdef SOLARIS
- #include <inttypes.h>
- #else
- #endif

Safe C Caution I --- about case

The API name with case means insensitive

- strcasestr_s
- strcasecmp_s

Safe C Caution II– memset_s

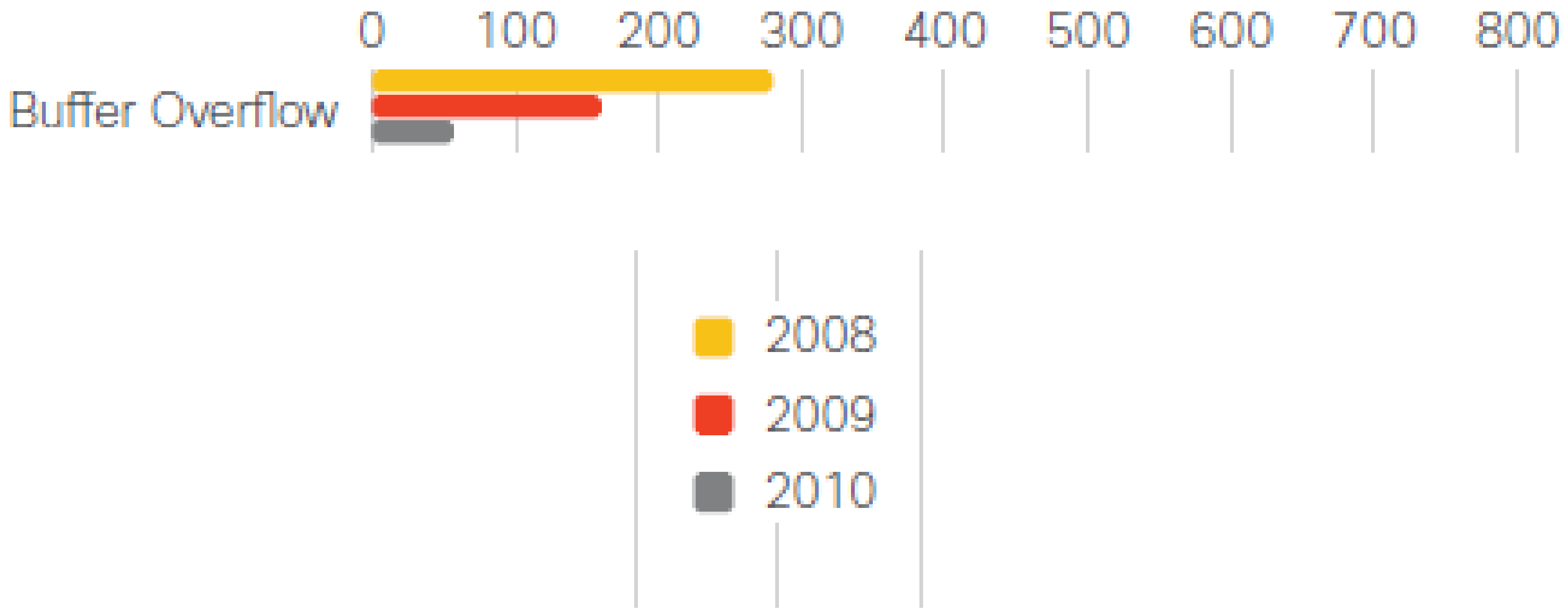
```
errno_t memset_s (void *dest, rsize_t len uint8_t value
        void*   memset(void *s int c , size_t n );
```

Use memzero_s to set memory to 0.

Safe C Caution III

Safe is based on the **correct** size of destination buffer.

The result of using Safe C



Summary & Conclusion