

## Re: compilation of ms crypto api program

**Source:** <http://www.derkeiler.com/Newsgroups/sci.crypt/2003-06/1068.html>

---

**From:** Henrick Hellström ([henrick.hellstrm\\_at\\_telia.com](mailto:henrick.hellstrm_at_telia.com))

**Date:** 06/22/03

Date: Sun, 22 Jun 2003 12:23:53 GMT

Ernst Lippe wrote:

- > *With all Pascal compiler that I have seen, even the first that*
- > *was developed by Wirth, it was possible to circumvent this by*
- > *using variants. Because in the original Pascal definition the*
- > *length of an array was part of its type this trick was commonly*
- > *used in many programs. But of course this breaks array bounds*
- > *checking. In this respect Java is much safer because it is*
- > *simply impossible to circumvent the array bound checks.*

I presume you are talking about variant record parts, such as:

type

  TMyArray16 = packed array [0..1] of 0..255;

  TMyArray32 = packed array [0..3] of 0..255;

var

  MyRecord: packed record

    case Integer of

      0: (Field1: TMyArray16);

      1: (Field2: TMyArray32);

    end;

  MyArray16: TMyArray16;

Given these declarations, the following statements are perfectly safe:

  MyRecord.Field2[0] := 0;

  MyRecord.Field2[1] := 1;

  MyRecord.Field2[2] := 2;

  MyRecord.Field2[3] := 3;

  MyArray16 := MyRecord.Field1;

MyArray16 will be equal to (0,1) and no array bounds have been broken.

The size of MyRecord will be equal to the size of the largest variant record part, and it is impossible to use it to access unallocated memory.

- > *It always amazes me how obsessed some crypto people are with*
- > *performance. Perhaps it is a bit naive, but I would expect that they*
- > *should be far more interested in security. Buffer overflows are*
- > *probably the most common source of security problems in software. They*

sci.crypt: Re: compilation of ms crypto api program

> *can be completely eliminated by using an appropriate language.*

No, buffer overruns are *\*not\** the most common source of security problems in software. It is a marginal problem generally, and occurs mostly in languages that e.g. allow dynamic allocation of stack memory. In those cases buffer overruns do occur, they will for the most part cause the software to crash badly long before any attacker will stand a chance to exploit them. There are lots of other security problems that are both much more common and run a much higher risk of being exploited before they are discovered.

> *Also, I believe that you are exaggerating the performance problems of  
> Java. First of all, interpreted languages in general have a constant  
> overhead compared with compiled languages. When your hardware becomes  
> faster, and it still does, this constant factor gets less  
> important. In many programs IO is the bottle-neck and that is not  
> dependent on your programming language. Second, modern JIT (Just in  
> Time) compilers can do very important optimizations, in many cases  
> there is no real difference for the end-user between Java programs and  
> those written in compiled languages.*

Sure, performance might be a marginal problem if you can afford buying faster hardware. I would say memory management and garbage collection is a much greater problem for the pov of security. I don't like the idea not being able to tell if a statement will cause key material to be copied to memory locations outside of my control.