April 16, 2008

WG14 N1301: Dealing With Pointer Subterfuge

Arjun Bijanki

Issue: storing a function pointer represents a security risk.

Problem: A hacker exploiting a vulnerability in a program could potentially overwrite the function pointer and thereby hijack the process when the function is called. Note that this attack can even occur on systems where the stack is not executable.

Mitigation: Instead of storing a function pointer, store an encrypted version of the pointer. An attacker would need to break the encryption in order to redirect the pointer to other code. This is similar to what's recommended when dealing with other sensitive data (e.g. passwords).

Microsoft Windows has a pair of APIs (EncodePointer / DecodePointer) that facilitate this, and are used by Visual C++'s C runtime libraries.

Note that this does not prevent buffer overruns or arbitrary-memory-write attacks, but it does make such attacks more difficult to exploit.

Example:

```
#include<windows.h>
#include<stdio.h>

typedef void(*PF)();

void myfunc()
{
      puts("myfunc called");
}

PF pf = NULL;

int main()
{
      PF pf_temp;


      /* suppose we need to store a function pointer; encode it for
         better security */
      pf = (PF)EncodePointer((void*)&myfunc);


      /*  ... call out to other code ...  */


      /* retrieve the function pointer */
      pf_temp = (PF)DecodePointer((void*)pf);
```

```
    /* call the function; if an attacker managed to overwrite 'pf',
       this call is likely to just cause a crash instead of execute
       the attacker's intended payload */
    pf_temp();
}
```