Security: Don't forget to initialize the stuff you don't care about

devblogs.microsoft.com/oldnewthing/20060703-00

July 3, 2006

Raymond Chen

Lost in excitement of privilege escalation vulnerabilities is the simple information disclosure through missing garbage initialization. Everybody should by now be familiar with the use of the `SecureZeroMemory` function to ensure that buffers that used to contain sensitive information are erased, but you also have to zero out buffers before you write their contents to another location. Consider, for example, the following binary format:

```
struct FILEHEADER {
    DWORD dwMagic;
    DWORD dwVersion;
    WCHAR wszComment[256];
    DWORD cbData;
    // followed by cbData bytes of data
};
```

Code that writes out one of these files might go like this:

```
BOOL SaveToFile(HANDLE hFile, LPCWSTR pszComment,
                DWORD cbData, const BYTE *pbData)
{
  DWORD cbWritten;
  FILEHEADER fh;
  fh.dwMagic = FILE_MAGICNUMBER;
  fh.dwVersion = FILE_CURRENTVERSION;
  fh.cbData = cbData;
  return SUCCEEDED(StringCchCopyW(
             fh.wszComment, 256, pszComment)) &&
         WriteFile(hFile, &fh, sizeof(fh), &cbWritten, NULL) &&
         cbWritten == sizeof(fh) &&
         WriteFile(hFile, pbData, cbData, &cbWritten, NULL) &&
         cbWritten == cbData;
}
```

Do you see the security bug?

If the comment is shorter than 255 characters, then the bytes after the terminating null consist of uninitialized stack garbage. That stack garbage might contain interesting information that you didn't intend to leak into the file. Sure, it won't contain information that you already recognized as highly-sensitive, such as passwords, but it still might contain information that, while less sensitive, still would be valuable to somebody looking for it. For example, depending on where the compiler decided to put local variables, you might leak an account name into those unused bytes.

I'm told that one company's networking software from a long time ago had a bug just like this one. They used a very advanced "change password" algorithm, the details of which are not important. The design was that only heavily encrypted data was transmitted on the wire. That way, somebody who sat on the network and captured packets wouldn't see anything of value. Except that they had a bug in their client: When it sent the encrypted password to the server, it forgot to null out the unused bytes in the "change password" packet. And in those unused bytes were, you guessed it, a copy of the password in plain text.

Raymond Chen

Follow