

Re: How to develop a random number generation device

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2007-09/msg02313.html>

- *From:* David Brown <david.brown@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 12 Sep 2007 23:46:24 +0200
-

Nobody wrote:

On Wed, 12 Sep 2007 13:42:56 +0000, No Spam wrote:

Nothing the OS does can prevent machine code from
overrunning a buffer.

That's not true. Many operating systems are by design, immune to buffer
over-runs modifying unrelated code.

The issue isn't about modifying code, related or otherwise. It's about
either injecting new code or executing existing code with
attacker-supplied data.

This isn't about protecting one process from another, but about protecting
a process from itself. Most of the existing mechanisms for mitigating
buffer overruns are implemented in either the compiler or libraries. The
only OS-level mechanisms (things that work on any executable, however it
was built) involve making it harder to exploit an overrun (e.g.
randomising memory locations) rather than actually preventing the overrun.

Given that:

- a) this would make Windows totally incompatible with most
existing
software, and

Did you mean to write "nothing the *Windows* OS does can prevent
machine code from overrunning a buffer?"

No, the issues apply to any OS. But binary compatibility is much more
important for Windows (and Mac) than for Linux.

If you try to run a 5-year old Linux binary on a current distribution,

Re: How to develop a random number generation device

you'll probably find that a lot of the interfaces on which it depends have either disappeared or have changed in an incompatible manner. Lack of a stable ABI is a simple fact of life on Linux.

Binary compatibility for the user-level API is extremely important for Linux, and the API is very stable – new API's and system calls are added, but existing ones are seldom changed or removed, and never without very good reason. Most five year old Linux binaries will run fine on the same architecture as they were build for (assuming you have the right libraries) – the only API calls that have changed are very seldom used.

I think you are confusing the issue with the device driver binary API, as used by device drivers internal to the kernel. There the API is very explicitly flexible, and may be changed whenever it is convenient (the *source* level API compatibility is high – no one wants to have to make extra source level changes without reason. But if all that's needed is a recompile, that's okay).

.