

Re: How to develop a random number generation device

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2007-09/msg03282.html>

- *From:* Nobody <nobody@xxxxxxxxxxxx>
 - *Date:* Mon, 17 Sep 2007 23:37:49 +0100
-

On Sat, 15 Sep 2007 17:06:31 +0000, Rich Grise wrote:

That doesn't address the issue, which was whether the OS can prevent buffer overruns.

With a hardware MMU, and software that can catch the exception, yes.

That still doesn't address the question of how you decide that a write operation has overrun its buffer; the details of where one buffer starts and another ends are unknown to the OS.

But it knows what chunks of memory it has allocated to a particular process. As long as it's in your own memory space, who cares if you overwrite/overrun your own buffers?

Doing so is the essence of a "buffer overrun exploit", one of the most common types of security vulnerability for code written in C/C++.

It allows a malicious user to make a program do something that it isn't supposed to do.

E.g. consider a program being run on a web server to process form input from a web page. If the program suffers from a buffer overrun flaw, simply sending the right data in a POST request can allow the attacker to execute arbitrary code on the web server.

Or a buffer overrun in a mail client could allow someone to run arbitrary code on the user's machine by sending them a specially crafted email.

This is one of the common ways that computer systems get "hacked".

Re: How to develop a random number generation device

Persuading a process to write outside of its allotted address space is harmless. The CPU will cause an exception and the OS will typically terminate the process. Even if it didn't, there's nothing there for it to damage. With modern hardware (e.g. 80286 and later running in protected mode), the address space of one process (or the OS kernel) simply isn't "visible" to another process.