

Re: gets() is dead

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-05/msg02046.html

- *From:* Flash Gordon <spam@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 21 May 2007 08:05:43 +0100
-

Tor Rustad wrote, On 21/05/07 01:47:

Flash Gordon wrote:

Tor Rustad wrote, On 20/05/07 16:03:

Why do so many worry about **trivial** thing like gets(), when they don't have a **clue** about writing secure programs?

The first step in writing a secure program is not to do things you know are insecure. Not using gets comes under this category. Even if

<snip>

The **first** step in building a secure system, is to design and analyze it **before** any programming has been started. That is the hard part, in my experience.

Failing to analyze and design (in my opinion you analyze the problem before you start the design) is also doing something you know is insecure, so that is also covered by my statement :-)

Now, after the spec has been implemented, time comes to testing and code audit. When doing audit, both manual inspection and static analysis should be done for safety-critical systems.

Agreed. Although secure and safety critical are independent attributes. For example, I have worked on safety critical SW where security was not a requirement (it was an embedded avionics system).

> Please name a single

Re: gets() is dead

security scanner, which don't warn about gets()!

I would not trust any which did not :-)

<snip>

How many here worry about making their code
time-independent?

Where that is an appropriate consideration (e.g. a login process) I will use a
tried and tested third party library from a source I have good reason to trust.
Far safer to use code from an expert than to try and get it correct yourself!

I can't afford to trust some software, just because a company have a
good reputation or something. I want proof.

Depending on the level of security I may or may not require proof. For the stuff I currently work on the
reputation of openssl is good enough, for other things I accept you would require proof.

Making time-dependent PIN or password check, would be a cardinal sin for a security
programmer. DPA (differential power analysis) is another example of side-channel attack.

I'm aware of this, although I've never worked on anything critical enough to worry about it.

or verify their calculations???

For security related calculations I will use trusted third-party SW that
undergoes testing of sufficient rigour.

There is a lot of snake-oil on the market.

Agreed. One does have to have proof that the company can be trusted, such as proper independent audits.

> One basic problem, is that

you can't really protect secrets in software, but need tamper-resistant hardware.

A powerful attack is fault injection, RSA can for example be broken after a single faulty
calculation.

Re: gets() is dead

Re: gets() is dead

There is more than one way to achieve this. Sometimes this can be achieved by having the HW in a room with sufficient physical security (where sufficient depends on the application).

How many have done threat analysis, and used formal methods to guard against sophisticated attacks?

If your threat analysis says you only have to defend against simple threats then you don't need to go too far in your protection. For example, if the system is entirely internal *and* the level of expertise of people on the inside is sufficiently low (builders, accountants etc) then the cost/benefits analysis says you do not need to go that far. That does not mean you should not avoid the more easily avoidable problems.

If the value you want to protect is low, well then of course you don't use a TCB (Trusting Computing Base), have guards, dual access control and video surveillance etc.

Depending on your definition of low value, I agree.

OTOH, for a bank, the insider threat is a *major* concern. However, banks want to make money, so they take calculated risks and in some cases, higher risks than they are aware of.

Yes, I fully agree.

In my case it really is builders and accountants who are generally not that knowledgeable about computing, and that factors in to the risk assessment. There is a lot of money involved, but the chances of anyone having the access and knowledge to mount a sophisticated attack without also having sufficient permissions to not need to attack are really small.

Unless ppl have read/written a protection profile, you are likely quite clueless in making a program/system secure IMO.

Unless you avoid the simple problems (which you may well do for all I know) then you are quite likely even more clueless.

Even a non-expert application programmer know about gets(), but they don't make secure C programs. You need a security professional to do that, and no, I have *never* located a gets() call in such code during audit.

Re: gets() is dead

I agree that for real security you need security professionals. Just as for safety critical applications you need safety experts.

Removal of gets(), will help clueless programmers, who are not building secure software anyway. For professionals, gets() is a non-issue IMO.

Removal of it means there is one less thing to learn not to do and one less thing for teachers to get wrong. I admit it will not actually be removed from the language for at least 20 years, but I don't see that as a reason not to start the process. Just as changing the law to require seatbelts be fitted in new cars was a good move even though it did not magically get seatbelts fitted to all cars.

--

Flash Gordon

.