

## Re: DJB's students release 44 \*nix software vulnerability advisories

*Source:* <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2004-12/0294.html>

---

*From:* Stephen Samuel ([samuel\\_at\\_bcgreen.com](mailto:samuel_at_bcgreen.com))

*Date:* 12/21/04

Date: Tue, 21 Dec 2004 11:39:49 -0800

To: "D. J. Bernstein" <[djb@cr.yp.to](mailto:djb@cr.yp.to)>

Side effects is my issue here.

It's a fine thing to encourage programmers to design their software well, and fix bugs quickly. I've got no problems with punishing programmers for bad code, but there are different issues to deal with in the classroom and out in the wild.

If you wanted to run a course where students are failed for having one bug in a 10,000 line project, they were aware of the standards going in, and given the training and tools such that the standard was achievable, not only wouldn't I have a problem with it — I'd encourage everybody I knew to hire anybody who passed the course. (at least, anybody I knew who gave a rat's ass about producing quality code).

When dealing with code out in the wild, however, you have two communities to consider — the programming community and the user community. Hmm. OK, three — there's also the cracking community.

We're now living in a world where the time from description of a bug and creation of a commercially viable exploit can approach zero. The 'commercially viable' exploits are viruses and worms used to generate zombie armies of tens of thousands of remote-controllable 'bot machines that can spread spam, launch DDOS exploits and do whatever else their controllers wish.

Out in the wild, there is a wide gamut of care about producing quality code and fixing bugs. At one extreme is Microsoft taking about 6 months to *\*NOT\** fix a security bug in WMP 9. (The fix is only available in WMP 10 released this week). That the programmer who reported the bug to MS would wait the 6 months for *\_any\_* real response from MS, I would agree with your probable determination that it is a disservice to the user community and does little more than reward MS for "trying to protect their shoddy security practices."

Near the other extreme are BSD coders working feverishly to fix a bug (and find any related errors) with *\_theoretical\_* security issues in

## SecurityFocus Bugtraq: Re: DJB's students release 44 \*nix software vulnerability advisories

their code in mere hours because the code was security related and they felt a very real responsibility to ensure that their community had secure code.

I would think that an optimal responsible disclosure policy should be designed to (as much as possible) punish the former while enabling the latter. Not all coders have achieved your standard of never, ever releasing buggy code at any time of their life. As much as we may wish otherwise, I doubt that that day will arrive soon either. A realistic disclosure policy needs to take into account the interests of the three communities I mentioned above.

I would suggest that a realistic disclosure policy would:

- 1) Discourage and punish shoddy security practices.
- 2) Enable responsible software maintainers to realistically respond to bugs.
- 3) Give the user community timely and usable information about bugs in the code that they're using.
- 4) Minimize the ability of the cracker community to exploit these bugs before 1 and 2 have

Your zero Time To Respond approach would address 1, but do nothing to satisfy 2–4. It doesn't so much punish bad programmers as it does encourage crackers to read your missives. Even the best of programmers have to sleep, and if you release your exploit just exploit just as the team/person responsible for the maintenance of a piece of software is going to sleep, crackers will have up to 8 hours to code and use your bug before the responsible programmer even wakes up to read the email about exploited code.

Even if rd party coders have managed to generate a fix, there won't be an official release of the fix until the responsible programmers have at least had an opportunity to vet the proposed fixes.

A reasonable and responsible disclosure policy should be to (unless there are signs that an exploit is already in use) allow responsible a realistic (if short) opportunity to fix the instant bug and audit their code for similar errors — thus at least allowing the \*possibility\* that a fix could be released at the same time as the announcement of the bug.

If a fix can be announced at the same time as the bug, users have an opportunity to install the fix while crackers are still crafting a commercial exploit.

I don't think that 24–48 hours (or even longer) is an unreasonable warning time for unexploited bugs. I'm not asking for time to 'be delusional', I'm asking for a reasonable amount of time for a responsible programmer to ensure that his/her user community is properly served and protected from the effects of the bugs.

SecurityFocus Bugtraq: Re: DJB's students release 44 \*nix software vulnerability advisories

6 months with no fix, on the other hand is obscene. I honestly doubt if companies like Microsoft will ever treat Security as a genuine responsibility as opposed to a PR issue. On the other hand, it is far less likely that they'll start responding reasonably if they don't even have a hope of being able to craft a fix before crackers have created a 'live' exploit.

D. J. Bernstein wrote:

> *Shu T. Messenger writes:*

>

>>*In each case, Professor Bernstein notified the author of the vulnerable package on Dec 15 via e-mail. This mail hit Bugtraq on the 16th, giving one day for vendors to provide fixes.*

>

>

> *Actually, I sent all of these notifications to the public securesoftware mailing list (<http://securesoftware.list.cr.yt.to>) at the same time that I sent them to the authors. It certainly wasn't my intention to give the authors an extra day of self-delusion.*

>

>

>>*Is the class on responsible disclosure next semester perhaps?*

Also posted: <http://www.bcgreen.com/comments/2004/bug-release.html>

--

Stephen Samuel +1(604)876-0426

samuel@bcgreen.com

<http://www.bcgreen.com/~samuel/>

Powerful committed communication. Transformation touching the jewel within each person and bringing it to light.