

Ten Years of Trustworthy Computing: Lessons Learned

Microsoft's Trustworthy Computing directive will be a decade old in January. Back in 2002, Bill Gates sent out an email to Microsoft employees, saying that security was the "highest priority" for the company.

He said in that email, "When we face a choice between adding features and resolving security issues, we need to choose security."

And that Microsoft has done. In the past decade, estimates are that the company has spent more than two billion dollars on software security—and accomplished a lot of impressive things. For instance, it trained all its developers on security and has gotten rid of numerous potential and real security problems. As a result, the company can show progress, for instance, by demonstrating that Vista had far fewer critical exploitable security issues after release than Windows XP did.

One of the highest-impact things Microsoft did was add protections to the library, compiler, and operating system. Two of the most effective protections they added were Data Execution Prevention (DEP)—which makes all writable memory nonexecutable, and process address space layout randomization (ASLR). These protections have made software exploitation of memory errors significantly more difficult.

Microsoft has also built its own security overlay for its software development process, which it calls

the Security Development Lifecycle (SDL). The company's engineers have written books about the SDL, and there are even people who can consult with other companies on implementing the process.

All this effort has produced great results for Microsoft—and has taught the industry quite a bit.

What We've Learned to Do ... and Not to Do

We already knew going into 2002 that it's both expensive and impossible to squash every single security bug, particularly in large code bases such as an operating system. Microsoft showed us that, with effort, we can make progress. We've also learned that the Microsoft approach hasn't been a hit with too many other companies. In my experience, companies find the SDL too expensive. It works for Microsoft because of its size, and because of the potential for brand damage if it's seen as the OS that's bad at security. But for most software companies, the cost of a full SDL implementation clearly outweighs the cost of alternative strategies. For many companies, alternative strategies means ignoring the problem until there's a disclosure, and then just patching the problems people

find. Doing more would be possible for such companies, but they don't want to spend the money without the return on investment.

For instance, let's look at training. For the sake of example, let's conservatively assume companies that want to train their staff on software security will spend US\$500 per employee (training only the developers, but spending more than \$500 per developer). Microsoft will spend \$30 million on its 60,000 employees, or approximately .04 percent of its \$70 billion of revenue. But when I worked at McAfee as a software executive, if we had spent the same amount for its then-6,000 employees, it would have cost more than three times as much percentage-wise. Similar economics applied to many of its activities—Microsoft was in a better position to scale. For instance, building secure development libraries can be done at a relatively fixed cost, which is a much smaller cost as a percentage of revenue the larger the company gets. Then, a good secure development library abstracts away work, and perhaps even saves time. The bigger company can make the same percentage investment, and will probably end up saving more coding time compared to smaller companies.

And the economics are even worse for very small companies. Consider a venture-backed startup with 40 employees that isn't yet making \$1 million per year. It would spend at least 2 percent of its revenue on training alone, and any time spent applying that train-



JOHN VIEGA
Editor in Chief

ing will cost more. A full SDL implementation would eat up far more than just the cost of training. Most start-ups of that size will choose to invest in product features, not security. And rightly so.

Eventually realizing that their full process wasn't for most other companies, Microsoft released a simplified version of the SDL in 2010, which seems to be gaining a bit more traction (see www.microsoft.com/download/en/details.aspx?displaylang=en&id=12379).

How Microsoft Improved Apple Security

Microsoft has also taught us that you don't have to spend to fix all the bugs to have good security. Particularly, the one-two punch of DEP and randomized process address spaces has made the many bugs remaining in its code far more difficult to exploit. In fact, we've learned that this fixed-cost technological investment is probably a better investment than training all developers.

To see evidence, we only have to look as far as Apple. Its lack of market share has kept OS X from being the target of a massive number of exploits. Apple has left a lot of exploits in its code, which is evident in the frequent security patches that remove dozens of significant vulnerabilities at a time.

It's still probably true today

that there are far more software vulnerabilities per thousand lines of code in Apple's system software than Microsoft's. Nonetheless, something big just changed.

In July, Apple released Lion, the newest version of its operating system. Lion includes the same kinds of protections Microsoft did years ago, with its own analogue to DEP as well as its own implementation of address space randomization. Plus, it has an excellent sandboxing mechanism for built-in and third-party applications. In fact, according to Ars Technica, Apple will soon require all applications sold through its App Store to be sandboxed.

As a result, Apple's Lion might just be safer than Windows 7, even though Apple got the security religion late in the game and spent a fraction of the money on security that Microsoft did. Of course, this probably wouldn't have been possible without Microsoft blazing the trail and showing what is cost effective.

I suspect Microsoft explicitly chose not to enforce the same level of sandboxing as Apple does, at least in part because it wants to give customers more control, for instance, letting them run very old legacy applications. But this teaches us that there's a big trade-off between control of the environment and security. Microsoft showed this with the Xbox, a product over which it

retained tight control of the environment. It not only was able to acquire many customers but also had a very good security track record, especially compared to commercial desktop operating systems.

And this is the model that Apple has followed, which has, so far, made iOS a much safer bet than Android. The Android store has already had a significant malware problem. There have been plenty of malicious apps that contain exploits placed on the Android store, letting the bad guys build up botnets of Android users. Apple, on the other hand, hasn't had many significant security issues outside of jailbroken phones, in which users explicitly circumvent Apple's security mechanisms on their own devices.

I'm by no means a shill for Microsoft—I've been a Mac user since before the Trusted Computing initiative, and I only use Microsoft for its Office applications. In fact, I believe that Microsoft doesn't always have the most secure products. But, it's also clear that it has taught the industry many great lessons over the past decade, mostly by positive example.

And for that, we should give Microsoft our thanks. I do.

cn Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

Silver Bullet Security Podcast

In-depth interviews with security gurus. Hosted by Gary McGraw.



www.computer.org/security/podcasts

*Also available at iTunes

Sponsored by **SECURITY & PRIVACY** digital