clavister ThoughtLeader

Open Heartbleed Surgery – Securing Against Further Vulnerabilities

David Sandin, product manager at Clavister looks at the implications of the Heartbleed bug and the use of open-source code libraries in vendors' security solutions.

There's a saying that's familiar to many: "When you assume, you make an ass of you and me." It was a series of assumptions that have led to the widely-reported problems caused by the Heartbleed bug recently. At the heart of the issue is a simple coding error – the type of error that any developer could make. But the error itself wasn't the real problem; it was the assumptions made by thousands of people globally that led to the global headlines and the rush to close off the vulnerability.

- **Assumption #1:** someone had double-checked and security tested the code before it was added to the OpenSSL software repository.
- **Assumption #2:** because open-source development has a community of programmers working together free of commercial imperatives, it leads to software with fewer bugs.
- **Assumption #3:** because OpenSSL is used by tens of thousands of websites worldwide, including some of the biggest online brands, and in a range of vendors' security solutions, it's fully tested, robust and secure.

With Heartbleed, all of these assumptions were proved wrong. Before I go further, let me be absolutely clear: I am not criticising or blaming open-source software development, nor am I criticising any of the individuals involved in developing, deploying or using OpenSSL. The world's most popular commercial software, worked on by huge development teams and used by hundreds of thousands of companies, is just as prone to serious, widespread bugs and vulnerabilities. If it wasn't, we wouldn't have Patch Tuesday every month. Mistakes happen, flaws get introduced.

But the current situation does highlight that effective Internet security has to be based on more than just people's assumptions, and the fact that thousands of other companies worldwide use the same security code on their websites and as part of their security solutions. There is no global Internet security task force that actively seeks and closes off these types of vulnerabilities when they are discovered. The blind rush to deploy OpenSSL – because everyone else was using it and, being open-source, it was cheap – played the major role in creating the scale and seriousness of the Heartbleed flaw. People trusted, but failed to verify that their trust was deserved.

Double jeopardy

There's also a more serious security issue to be considered. As I touched on briefly above, OpenSSL isn't just used by companies who wanted to enable security on their websites. The code had also been built into big-name vendors' networking and security software and solutions, including servers, routers, gateways, appliances and more. A huge number of these must also be updated, as the list at the Computer Emergency Response Team (CERT) website shows – creating a real headache for companies' IT teams that have to check their solutions and apply updates.

Why do commercial networking and security solutions include OpenSSL code, or other large open-source codebases? For much the same reasons as any other organisation, as I outlined above: it's cheap, widely used, helps to make solution development and adding new features faster, and is assumed – there's that word again – to be secure.

It seems that vendors using the code in their solutions didn't bother to check it before integrating it. As a result, many end-user organisations were deploying security solutions that used affected versions of the code, to protect applications and web services that were using the exactly same OpenSSL code, carrying the same vulnerability. This would potentially allow an attacker to walk straight through the security solution, and attack the application or service by simply exploiting the same vulnerability twice. It's like putting two combination locks on your bicycle, and setting both to use the same combination.

What's in the box?

It's not good security practice to use a large open-source library as part of a security solution without rigorous checking, because then you're trusting that another party has properly reviewed the code, and issued appropriate patches for it. It was recently announced that an open-source software group was creating a simpler, cleaner version of OpenSSL, and had already removed nearly 250,000 lines of code and content that wasn't needed.

If so many lines could be removed, how many more are completely irrelevant for use on a network security gateway or firewall? How many potential vulnerabilities are there in that code? The bigger the third-party codebase used, the greater the potential for a flaw that can be attacked. How could the vendors building the OpenSSL code into their solutions be sure that all the code was secure? Put simply, they weren't sure: they assumed.

Even vendors using OpenSSL in their solutions who claim that they are not vulnerable to Heartbleed, could simply be using older versions of OpenSSL that may not be affected by the Heartbleed problem. But that doesn't mean that the code has no other vulnerabilities. After all, it took over two years for Heartbleed to be discovered.

Security solutions need to be rigorously developed, tested and re-tested to ensure any vulnerabilities are removed. They should not include, or rely on off-the-shelf code which has not been verified as being secure, no matter how appealing it is to try and accelerate the development of products. Security must be about trust, founded on a solid technical basis. And that applies to consumers, major websites, and IT security vendors alike. The Internet already has enough threats and vulnerabilities, and we don't need any more being introduced by dangerous assumptions.

More From Clavister

For more though leader articles and information about Clavister products, visit **www.clavister.com**.

Clavister AB, Sjögatan 6 J, SE-891 60 Örnsköldsvik, Sweden
Phone: +46 (0)660 29 92 00 | Fax: +46 (0)660 122 50 | Web: www.clavister.com