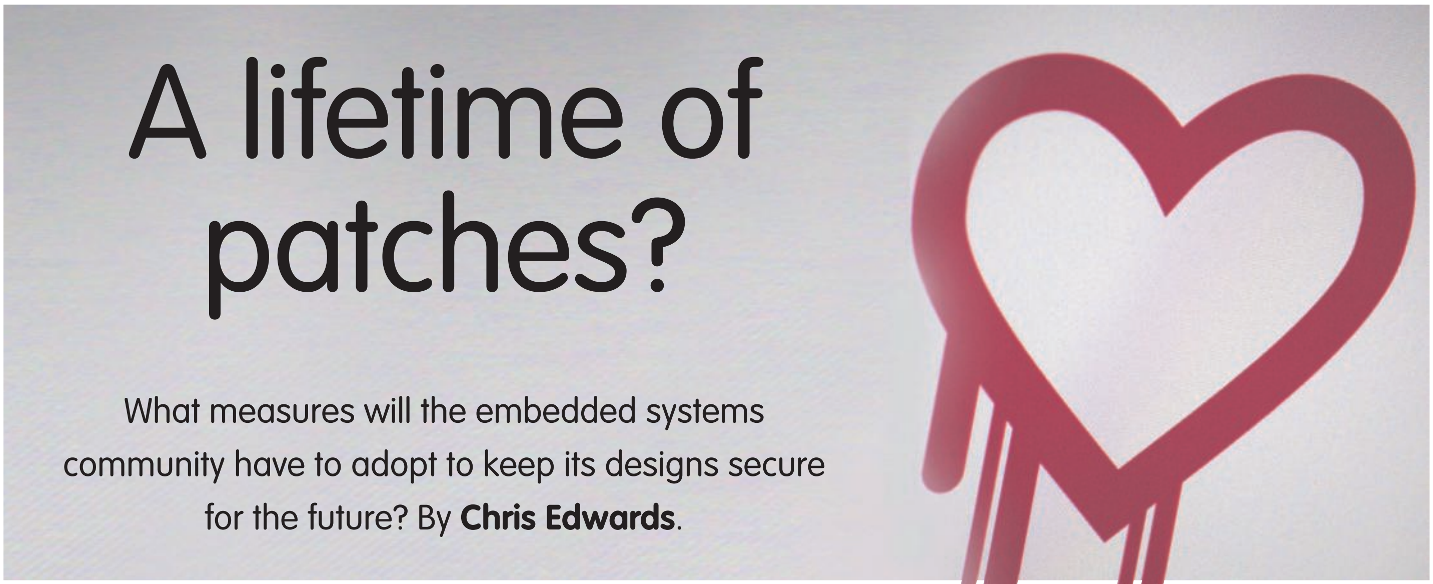

A lifetime of patches?

What measures will the embedded systems community have to adopt to keep its designs secure for the future? By **Chris Edwards**.

In the wake of the disclosure of the Heartbleed bug that punched a hole in the protocol for encrypting web page transfers, NXP's chief security technologist, Mathias Wagner, looked at the IP camera he had just bought to see whether it was afflicted and, if so, what he could do to update the firmware.

Not only was the camera affected, Wagner says there was no way to upgrade the software to plug the hole in the Transaction Layer Security (TLS) implementation. "In today's environment, you need to be able to update these devices. But the ecosystem is not prepared for that," he says.

As the trend to connect embedded systems to the internet takes hold, hackers are finding more ways to reach into embedded systems without having to get near them. ARM's CTO Mike Muller argues the ability to patch vulnerabilities will be necessary.

"As a designer, you have to find all the flaws in the system; the attacker only one. You will fail. You have to plan for that failure. You have to build systems on the assumption they will get hacked. And you have to be able to refresh them," Muller says.

One reason for the need to patch embedded systems is, largely, the increased use of off the shelf software developed initially for the desktop: an environment where regular security updates have become not just accepted, but expected. This is where the lifecycles of embedded products and of 'borrowed' software can come into conflict.

While hacking into the Tesla Model S, Cloudflare principal security researcher Marc Rogers noticed the carmaker has already encountered the problem. At the heart of the Model S' instrument cluster is an nVidia Tegra processor running Ubuntu, but the lead times involved with design and verification meant the operating system and the software behind its browser are now considered obsolete. "The browser is really old now. It's stuck on an old version of Webkit, so it's got a ton of vulnerabilities," Rogers explained at the first seminar organised by the NMI's Internet of Things Security Foundation (IoTSF).

Tesla would need to patch these vulnerabilities itself instead of relying on the open source community, which has since moved on to newer versions of Ubuntu and Webkit.

"There is a clear mismatch between the life of some things and the

The operating system and software behind some of the features of the Tesla Model S are now considered obsolete

Although the Tesla Model S might have a lifespan of 20 years, some of the software on which it relies will only be supported for a fraction of that time

software they support. Ubuntu has five year support; the Model S has a possible life of 20 years," Rogers says.

Specialist embedded suppliers, such as Green Hills Software (GHS), say they are willing to plan for long life cycles, which may encourage companies to consider using their software, instead of allocating resource to otherwise dormant open-source code branches. Chipmakers are joining in: NXP is now using a maintenance business model so that it can handle long-term support for the software it develops for its silicon.

Avoiding enterprise style problems

With the benefit of being able to see what has happened in the desktop and server environment, the industry does not have to accept the same problems as the enterprise software world. Not everything should be designed with the idea that patching can keep devices secure; not least because there has to be some reliable code in there that can ensure that patches themselves can be applied securely. "If you've found a hole in your software and the hole is in your patching mechanism, you are in trouble," says Dave Hughes, CEO of HCC Embedded.

Chris Smith, vp of marketing at GHS, says: "There are plenty of industry sectors where software has to be architected correctly, designed to be safe and secure and designed not to touched."

Hughes adds: "These are people who have very well structured development processes. Using them, it is entirely possible to write secure software. There have been no major failures of the underlying protocols in recent years at all: TLS works. The failures have been down to poor software implementation or poor management of passwords. In terms of requirements, none of that comes down to patching: it's about doing a proper implementation."

Recognising that it is difficult to prove secure behaviour for all code, some OS suppliers have developed separation kernels, which can stop vulnerable software accessing parts that need to remain secure. In the coming years, MCUs from suppliers such as NXP will incorporate ARM's Trustzone-M, that adds hardware support for separation kernels and hypervisors, even to low end devices.

Smith says: "The enterprise space has, to some extent, had to embrace this patch based philosophy because it hasn't had a separation architecture to depend on. Embedded can be different – if you plan it. You can stop and think about the architecture and deal with these issues differently. Assuming that it will be insecure because you have a flat memory model? No, you don't have to do that."

Where patches are needed, embedded systems suppliers need to ensure they are authenticated and installed correctly. Andrew Girson, CEO of training company Barr Group, says: "We've seen a lot of companies expose themselves to more security holes because their upgrade process is not secure."

Organisations are beginning to develop guidelines for patching. Among them is the IoTSF, whose chairman John Haine says it aims to publish guidelines on how to patch and update constrained devices later this year.

Greg Rudy, director of business development for GHS subsidiary Integrity Security Services, says remote updates need to perform several levels of verification. The simplest, often performed using hashing techniques, is to check that data in the download image does not contain any alterations. "Then you need to check it's the proper code coming from the manufacturer. We typically do that through asymmetric cryptography and digital signing," he explains.

Running out of memory

A major problem is that memory constrained devices may not have enough space to hold the running software and its replacement. Dealing with this means being able to apply a series of small patches in turn. "This approach gets more complex when you bring security into the picture. With each new piece of code, you need a new signature.

"From what I've seen, the industry has the ability to pull apart the image and download only the pieces that are needed. That's done, as is signing code and pushing signed chunks of code across a network. It's the interleaving of those two that's currently under development."

Remote updates will not be the default position for some industries; Rudy says medical companies, for example, are moving towards updates being made only through a physical connection.

Manufacturers also need to think before reacting to every declared vulnerability, says Hughes. "Things are going to get broken, but you need to figure out the real world effect. A lot of hacks just provide a proof of concept and the danger is exaggerated. You have to look at the context: how could that hack affect people in real life?"

In an environment where remote hacking is inevitable, companies need to pay more attention to security. The ability to patch vulnerabilities will be useful, but more important is an infrastructure that protects the most sensitive parts of the target system.

Medical companies are moving towards software updates being made only through a physical connection