The Dead Software Foundation

We are writing to comment on Docket No. 180103005–8005–01, "Promoting Stakeholder Action Against Botnets and Other Automated Threats."

This comment focuses on goals, roadmaps and further activities surrounding sustainability. Sustainability is listed as Goal 1 of the draft; and “all stages of the lifecycle” are listed as a principal theme. The document must thus grapple with the idea of software with a 30-year lifecycle,[1] and we have no idea how to do this.

In 1988, original 512K Macintosh computers were still operational. MS-DOS 4 was released. People got online with Compuserve, Delphi, or GEnie. John Markoff had to explain what the Internet was when he wrote an article about a worm[2] which infected Sun and DEC workstations. And the furnace which warms me today was built, delivered, and installed.

The draft report must at least introduce, and ideally, grapple with, the hard problems of 30-year software, which presents unsolved software, economic, legal and policy problems. But the internet of things will include things, including appliances and vehicles, with a 30-year lifespan. That’s 50% longer than the now average 20-year age of the companies that make up the S&P 500.[3] The Morris worm infected Sun and DEC equipment. Both companies have been acquired, and their product lines are effectively dead. Recently, Logitech chose to replace all of its Harmony Link Hubs, rather than continue to maintain them.[4]

We need to start thinking about them, and considering the operation of a Dead Software Foundation in 2050 is a helpful lens. A Dead Software Foundation (DSF) would ensure that dead software is only mostly dead, but that there are developers who will backport fixes for security issues if development is ongoing, and develop fixes for new classes or families of bugs as needed.

The most obvious alternative would be to do nothing. That means we will have 30-year-old unmaintained software, which is even less attractive.

This raises immediate and important questions, including, but not limited to:

---

[1] 30 years may or may not be the right span, but it is usefully provocative.

[2] https://www.nytimes.com/times-insider/2014/08/06/1988-the-internet-comes-down-with-a-virus/

[3] https://www.cnbc.com/2017/08/24/technology-killing-off-corporations-average-lifespan-of-company-under-20-years.html

[4] https://www.wired.com/story/logitech-giving-harmony-link-owners-a-free-harmony-hub/ . The story explicitly notes “the small install base…and the work required to maintain it.”

- Who would contribute software, when and why? For example, if Microsoft contributed the source to Windows XP and security patches were made available, perhaps fewer people would buy Windows 10, or Microsoft might be concerned about that. If Linux without systemd were under maintenance, how does that effect the development of new systems? What happens if the Acme Furnace company keeps using Linux 2.0 kernels?
    - Dan Geer has suggested that abandoned software be open sourced[5], but open source is insufficient, we need a mechanism for fixes, and if a company enters bankruptcy, its software will be an asset owed to creditors, which is in conflict with contribution to such an effort.
    - Software is insufficient as a contribution. It is useless without the build tools, environment, and platforms needed to transform software into its final executable form of a program or firmware image.
- There is also the flipside of such a question, what software would a foundation accept? For example, there was an 11 line javascript program, "left-pad" whose removal broke a great many NPM-dependent packages.[6] Without scope constraints, a DSF would be unable to have staff familiar with its code. It may not just be software; it may be domains, or even service of MUD files.[7]

- Who pays? Those who leave software behind may be bankrupt or open source projects. Is software maintenance a public good? If so, and if governments pay, there are questions of who gets the jobs (is it ok for the American public to pay less expensive programmers in China?) What happens to knowledge of unfixed vulnerabilities? The US Vulnerabilities Equities Process and the Chinese CNNVD are already sources of distrust.[8]

- How do updates get to normal people? It's not enough to patch a library; in many cases someone may need to build new firmware, test it and install it.

- How does the DSF keep staff employed long term to avoid knowledge churn? 30 years is over half of a working lifetime.

All of these questions may seem to lead to the conclusion that a Dead Software Foundation is too hard. It may well be. But don't call me to complain that your neighbor's furnace has put randomware on your door lock and you can't get in. I'd be sympathetic, but I can't handle the North Korean propaganda the darn voice-imitating malware keeps injecting into our conversation.

---

[5] http://geer.tinho.net/ieee/ieee.sp.geer.1307.pdf
[6] https://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/
[7] Manufacturer Usage Description, an emerging protocol for constraining the network connections of a lightbulb or other internet enabled thing. While the protocol is emerging, in 30 years, it will be obstelete, along with the HTTP connections on which it relies.
[8] https://www.recordedfuture.com/chinese-mss-vulnerability-influence/

Immediate and short-term steps could include:

- Commerce to convene industry and academia to discuss how to maintain software over 30 years.
- NSF, DARPA, or others to fund research into the scientific and engineering challenges of maintaining today's software for 30 years.

We urge you to consider these issues of long-term sustainability.

Adam Shostack
President, Shostack & Associates
adam@shostack.org/917-391-2168

Nicko van Sommeren
CTO, Core Infrastructure Initiative

(Affiliations are listed for identification purposes only)