to: NTIA
from: Dan Geer, P.O. Box 235, Cornersville TN 37047
date: 5 February 2018
re: Docket 80103005-8005-01
cc: file

Software is unconstrained by any "wearing out" process. This means that software already in place can last until such time that the hardware it requires simply dies in an era when it cannot be replaced. In other words, installed software has a very long tail — which brings us to the point: In the vocabulary of your Goal #1, for software to be sustainably secure it must be adaptable. We will now discuss the implications of that requirement.

There is no need to re-prove that we are accumulating software security flaws.[1] There is no need to re-demonstrate that the need to find and fix flaws is serious enough that software vendors are heavily augmenting their own regular ground troops with soldiers of fortune.[2] There is no need to re-calculate whether the compound annual growth rate of the Internet is exactly 35%, the implication is clear.[3]

We are putting software into the field at massive rates (as a word, "massive" is not good enough, but it will have to do). We therefore come to a fork in the road: In one direction is hardware that is *designed* to fail in a fixed time — so as to force new hardware and by way of that new software. In the other direction is software that has a remote management interface — so as to ensure the software can be strengthened should that need to be done. And we have to do the one or the other; the combination of unfixable and immortal is anathema.

Commercial and non-commercial suppliers of software know this only too well, yet each and severally they abandon their unfixable products by the side of the information highway. The cynic might say that this is to sell more hardware, but besides being able to observe it with every bankruptcy, with every merger, with every product "end of life" letter, we can also see it on the net: A non-negligible fraction of Internet backbone traffic cannot be identified by protocol, i.e., it has no provenance. Intentionally obscure traffic may as easily be heroic freedom fighters posting unexpurgated calls to arms as it can be paedophiles, but it could just be junk traffic — traffic whose emitter is on auto-pilot but whose purpose is long defunct. Yet with Qualcomm's Swarm Lab at UC Berkeley predicting 1000 radios per human by 2025 and Pete Diamandis' book _Abundance_ calling for 45x10^12 networked sensors by 2035, we have to anticipate that we won't be able to hear or find all the devices out there even if they are not purposefully configured to be unfindable.

What happens when a single vendor abandons, let's say, 10 radios per human — just 10 out of a 1000? That is the issue. If I abandon a car on the street, then eventually someone will be able to claim title. If I abandon a bank account, then the State will eventually seize it. If I abandon real estate by failing to remedy a trespass, then in the fullness of time adverse possession takes over. If I don't use my trademark, then my rights go over to those who use what was and could have remained mine. If I abandon my spouse and/or children, then everyone is taxed to remedy my actions. If I abandon a patent application, then after a date certain its teaching passes over to the rest of you. If I abandon my hold on the confidentiality of data such as by publishing it, then that data passes over to the commonweal not to return. If I abandon my storage locker, then it will be lost to me and may end up on reality TV. The list is all but endless because it covers everything. Except for software.

We learned the hard way about the downside effects of abandoned industrial facilities, whether commercial or governmental. Applying those lessons learned to the software arena requires long range thinking. There can be no effective policy but this one: If Company X abandons a code base, then that code base has to be seized on behalf of the public interest, and the company abandoning that product, that code base, had damned well better still have the build environment that made the software in the first place because without the build environment the unexpurgated source code is just another form of toxic waste — mineable for nuggets of vulnerability but of no constructive value save to those of hostile intent.

At the very least, the legal standard of merchantability must immediately be amended to require that build environments are preserved and documented well enough that a receiver can use them to make repairs the original vendor will no longer do. And makers of software must put aside not only those constructive tools but also some meaningful down payment on the means to support their products, no different than a reserve for decommisioning a reactor or for revegetating a pit mine. In the same way that the public interest requires local governments' tax liens to have primacy at the time of any change in a property's use or ownership, the assets of the vendor dropping support have to be exposed to seizure should the vendor have failed to meaningfully preserve the build environment of fielded systems.

We have a long tradition in this country of kicking the can down the street in conformance with what I know to call the Four Verities of Government:

> Important ideas are rarely exciting.
> Exciting ideas are rarely important.
> Not every problem has a good solution.
> Every solution comes with side effects.

Add to that that while all politics is local all technology is global. Yes, we have to give up on some amount of optimality and efficiency if we are to have robustness and resilience, and we only get the latter if we require, truly require, life-cycle costing of software dependence, a fact made pressingly urgent by the rate of deployment of new sofware and new devices containing software. There is no doubt we've already lost control at some level;[4] everyone of a certain age has heard or said "We don't know what this code does but if we touch it everything falls apart so DON'T TOUCH IT." The entire Y2K episode was about this. But with a trillion instantiations of software, this may well be our last chance to retain self-determination.

---

[1] "Security Debt" geer.tinho.net/fgm/fgm.geer.1308.pdf

[2] "2018 Bug Bounty List" www.bugcrowd.com/bug-bounty-list/

[3] "Implications of the IoT" geer.tinho.net/fgm/fgm.geer.1612.pdf

[4] "Replace Your Exploit-Ridden Firmware..." www.youtube.com/watch?v=iffTJ1vPCSo

[illegible]