West Texas Cyber Security Consortium

Heartbleed as Metaphor

By Dan Geer

Monday, April 21, 2014 at 1:30 PM

I begin with a paragraph from Wikipedia:

> Self-organized criticality is one of a number of important discoveries made in statistical physics and related fields over the latter half of the 20th century, discoveries which relate particularly to the study of complexity in nature. For example, the study of cellular automata, from the early discoveries of Stanislaw Ulam and John von Neumann through to John Conway's Game of Life and the extensive work of Stephen Wolfram, made it clear that complexity could be generated as an emergent feature of extended systems with simple local interactions. Over a similar period of time, Benoît Mandelbrot's large body of work on fractals showed that much complexity in nature could be described by certain ubiquitous mathematical laws, while the extensive study of phase transitions carried out in the 1960s and 1970s showed how scale invariant phenomena such as fractals and power laws emerged at the critical point between phases.

That may or may not leave you cold. I begin with those lines because they say that complexity in the large can arise from locally simple things. I begin with those lines because the chief enemy of security is complexity. I begin with those lines because it explains why it is that we humans build systems that we can't then administer.

In your bones, you know all that. You also know that Nature teaches that where there is enough prey, there will be predators. Nature teaches that when a predator gains an advantage, it will consume prey as fast as it can metabolize them, until the prey are too few to sustain the predators' numbers. Nature teaches that monocultures are so unnatural as to require constant intervention to maintain.

How might this inform policy? A worked example seems the best answer to that question.

Recent headlines have been all about a coding error in a core Internet protocol that got named "Heartbleed." It is serious. It was hiding in plain view. If it wasn't exploited before its announcement, it most certainly has been after. It is hard to fix.

That Heartbleed is hard to fix is not because there is some technical difficulty in fixing a single line of code; it is hard to fix operationally — the error is so widely deployed that removing its effects will take time. Whether such a simple error could have been detected before it was so widely deployed is being debated with all the vigor of stiff-necked 20/20 hindsight.

When deployment is wide enough, it takes on the misfeatures of monoculture. Heartbleed is instructive; its deployment was not wide enough to be called an Internet-scale monoculture and yet the costs are substantial. What if Heartbleed had been a thoroughgoing monoculture, a flaw that affected not just the server side of a fractional share of merchants but every client as well?

Only monocultures enable Internet-scale failure; all other failures are merely local tragedies. For policymakers, the only aspect of monoculture that matters is that monocultures are the *sine qua non* of mass exploitation. In the language of statistics, this is "common mode failure," and it is caused by underappreciated mutual dependence. Here is the National Institute of Standards and Technology(NIST):

> A common-mode failure results from a single fault (or fault set). Computer systems are vulnerable to common-mode resource failures if they rely on a single source of power, cooling, or I/O. A more insidious source of common-mode failures is a design fault that causes redundant copies of the same software process to fail under identical conditions.

That last part — that "[a] more insidious source of common-mode failures is a design fault that causes redundant copies of the same software process to fail under identical conditions" — is exactly what monoculture invites and exactly what can be masked by complexity. Why? Because complexity ensures hidden levels of mutual dependence. In an Internet crowded with important parts of daily life, the chance of common mode failure is no idle worry — it is the sum of all worries.

Which brings us to critical infrastructure and the interconnection between critical infrastructures by way of the Internet. For the purpose of this essay, I will use the definition found in Presidential Decision Directive 63, issued by then-President Clinton:

> Critical infrastructures are those physical and cyber-based systems essential to the minimum operations of the economy and government.

The Internet, *per se*, was designed for resistance to random faults; it was not designed for resistance to targeted faults. In point of fact, you cannot have both resistance to targeted faults and resistance to random faults. But what monoculture does is to decrease resistance to targeted faults.

The critical infrastructure's monoculture question was once centered on Microsoft Windows. No more. The critical infrastructure's monoculture problem, and hence its exposure to common mode risk, is now small devices and the chips which run them. As the monocultures build, they do so in ever more pervasive, ever smaller packages, in ever less noticeable roles. The avenues to common mode failure proliferate. While it may not be in the self-interest of the Ukrainian mob to silence the Internet, nation states in conflict may make different choices. As Stuxnet showed, even exceptionally precise targeting only delays the eventual spread of

collateral damage. Monoculture leads to common mode failure and thereby complicates disambiguating hostile actions from industrial accidents.

One example of an effective monoculture, albeit within a domain that is almost but not quite Internet-scale, is the home and small business router market. Most on offer today are years out of date in software terms and there is *no* upgrade path. Those routers can be taken over remotely and how to do so requires low skill. That they have been taken over does not diminish their usefulness to their owner nor is that takeover visible to their owner. The commandeered routers can be used immediately, which may be the case with an ongoing banking fraud now playing in Brazil, or they can be staged as a weapon for tomorrow, which may describe the worm called TheMoon that is now working its way through such devices. The router situation is as touchy as a gasoline spill in an enclosed shopping mall.

The Heartbleed problem can be blamed on complexity; all Internet standards become festooned with complicating option sets that no one person can know in their entirety. The Heartbleed problem can be blamed on insufficient investment; safety review for open source code is rarely funded, nor sustainable when it is. The Heartbleed problem can be blamed on poor planning; wide deployment within critical functions but without any repair regime.

There seem to be three ways out of this dilemma.

* The first would be to damp down new installations of pretty much anything until we can get a handle on what our situation really is. When Microsoft hit the wall, that is exactly what (to their credit) they did—they took a significant breather from new deployment and started over wherever necessary. Electronic Health Records and the Smart Grid are two obvious candidates for this as each are at the stage of "last chance to get it right without breakage."

* The second is to make new digital installations prove in some strong sense that they cannot be harmful before deployment is allowed. Clinical trials for drugs follow this model. Applying such a model will be hard as so much operational security involves how something is run as much as how it is built. The FAA will tell you that the reason that airplane failures are singletons is that suppliers have to prove that they won't make the same mistake a second time.

* The third is to make recovery mechanisms so fast that you might not even notice the failure. (Think electrical supply for the hospital operating room.)

Put differently, the three alternatives are (1) to stand down risks already in the system, (2) to drive the mean time between failures to infinity, or (3) to drive the mean time to repair failures to zero. The first one is the easiest technically and the hardest politically. The second one requires a discipline that will not come without some threat of force, and it requires giving up on "innovation uber alles." The third may be circular insofar as instant recovery requires hands-off automaticity, meaning no people in the loop.

What is different about cyber security compared to, say, flight safety is that digital regimes have sentient opponents, and you cannot calculate product fail times when sentient opponents are the risk rather than metal fatigue. Pedantic though it may sound, any product with a sentient opponent is a security product, and all security products are dual use.

There are two areas of research that hold special promise. One is in the area of proof, what is known as "language theoretic security"[LS] in particular. LANGSEC adoption will require significant rework of nearly any and all of the Internet's underpinnings where one party must take input from another. Such re-engineering will require time, but it is precisely on point insofar as remote takeover means attacker input being processed by a target.

The other, known as the "honeymoon effect," mirrors co-evolution between prey and predator, and prescribes that update of deployed software code bases occur at whatever rate exceeds the ability of its (sentient) opponents to retool their methods. Such speed would require a remote management regime unlike what is generally deployed now.

All of this brings us to the question of open versus closed source. While there are valid arguments all around, if one assumes that failures will happen, then open source is to be preferred insofar as in that case, (the collective) we can learn something from said failures. That being so, then the more one depends on XYZ the more one needs XYZ to be open source, along with the build environment through which it passes. The latter can be analogized as how the airworthiness certificate a jet engine gets is not for the engine as a hunk of metal but for the design and manufacturing processes that produce that hunk of metal.

It may be that such rigor is not just for things that have already been recognized as critical infrastructure. Why? Because it is impossible to ascertain at the time of introduction whether something new will or will not go to scale.

Heartbleed is getting its fifteen minutes of fame, but what may matter most is that so much of what is being deployed now is in the embedded systems space — network-capable microcontrollers inside everything that has a power cord or a fuel tank. No one watches these and they are treated as if immortal. They have no remote management capability. There is not even a guarantee that their maker knows with precision what went into any one of them after the model year is over. The option suggested by the honeymoon effect is thus impossible, so the longer lived the devices really are, the surer it will be that they will be hijacked within their lifetime. Their manufacturers may die before they do, a kind of unwanted legacy much akin to space junk and Superfund sites. BBC science reporting has already said much the same thing.

To repeat, Heartbleed is a common mode failure. We would not know about it were it not open source (Good). That it is open source has been shown to be no talisman against error (Sad). Because errors are statistical while exploitation is not, either errors must be stamped out (which can only result in dampening the rate of innovation and rewarding corporate bigness) or that which is relied upon must be field upgradable (Real

Politik). If the device is field upgradable, then it pays to regularly exercise that upgradability both to keep in fighting trim and to make the opponent suffer from the rapidity with which you change his target.

Suppliers that refuse both field upgradability and open source access to their products should be said to be in a kind of default by abandonment. Abandonment of anything else of value in our world has a regime wrapped around it that eventually allocates the abandoned car, house, bank account, or child to someone new. All of the technical and procedural fixes to the monoculture problem need that kind of backstop, *viz.*, if you abandon a code base in common use, it will be seized. That requires a kind of escrow we've never had in software and digital gizmos, but if we are to recover from the fragility we are building into our "digital life," it is time. I am glossing over the details to be sure, but in the same way that John Kennedy didn't describe what it would take to get to the moon.

So I say to policymakers, the piper will be paid. That payment can take the form of a continuous, quiet theft of the sort inflation imposes on savers but for which no legislature need be on the record. That payment can be to bring to digital goods what physical goods have long endured, a regime where either the maker empowers the user to fully take care of himself or the maker carries the responsibility for the user-level downsides of their technology. It cannot remain as it is. We are at the knee of the curve.

More if you want it. Lots more. Be in touch.

[This and other material on file under geer.tinho.net/pubs.]

Daniel E. Geer, Jr., Sc.D., serves as Chief Information Security Officer at In-Q-Tel, the strategic investment partner of the U.S. intelligence community, and has held C-level positions at six startups over the past two decades. Prior to that, he led systems development at MIT's Project Athena out of which came many of the underpinnings of today's Internet and, earlier still, worked in medical computing within Harvard's various teaching hospitals. He provides advice and counsel to numerous Federal agencies, and has been before Congress five times. Dr. Geer's degrees are in Biostatistics from the Harvard School of Public Health and in Electrical Engineering from MIT, and he has been honored with the Lifetime Achievement Award of the USENIX Association.

http://www.lawfareblog.com/2014/04/heartbleed-as-metaphor/