For Good Measure
Security Debt

DAN GEER AND CHRIS WYSOPAL

Dan Geer is the CISO for In-Q-Tel and a security researcher with a quantitative bent. He has a long history with the USENIX Association, including officer positions, program committees, etc.
dan@geer.org

Chris Wysopal, Veracode's CTO and co-founder, is responsible for the company's software security analysis capabilities. In 2008 he was named one of InfoWorld's Top 25 CTOs and one of the 100 most influential people in IT by eWeek. One of the original vulnerability researchers and a member of L0pht Heavy Industries, he is an author of L0phtCrack and Netcat for Windows, and is the lead author of *The Art of Software Security Testing* published by Addison-Wesley.
cwysopal@gmail.com

> Blessed are the young for they shall inherit the national debt.
>
> — *Herbert Hoover*

When you start a company, you take on financial debt so that you can reach your market in time. When you release a product, you take on technical debt, and for the same reason. Ward Cunningham talked about this in 1992 [1]:

> [I]mmature code may work fine and be completely acceptable to the customer, excess quantities will make a program unmasterable, leading to extreme specialization of programmers and finally an inflexible product. Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite…The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation.

One of the present authors proposed [2] that cyberinsecurity is the critical form of technical debt, if for no other reason that a bug is exercised by an accident but a vulnerability is exercised by an enemy. Consider your security vulnerabilities to be a debt note that has been purchased by someone who is out to get you—not only are you in debt, but the debt can be called at the most inconvenient time calculable.

Every software release is debt issuance, and vulnerabilities are common in fresh code [3], but as Clark et al. point out [4], you can roll your code often enough that the attackers can't keep up. (Google appears to roll Chrome every 3–5 days.) Rolling over a financial debt cheaply means life is good, unless and until there is a rate shock.

The problem with rolling over your security debt, however, is that you can soon have no idea what is going on. If you are a supplier, then you may choose to buy outside testing, that is to say you may choose to get your debt rated, but with sub-week release cycles it is not possible to test within cycle—test results are always for a now previous version. If you are a consumer, your test might be the most trivial of all tests, viz., whitelisting the hash taken from the supplier's golden master, but propagation time for the whitelist may well not keep up with the rate of issuance, just like a rating agency that can't even rubber stamp what the mortgage lender is issuing as fast as they are issuing it.

Let's say you've been rolling over your cyberinsecurity debt for long enough that you have a considerable debt overhang built into your products, or into your enterprise deployment of everything from Aardvarks to Zebras. Well, you can pay it down. Microsoft showed us how when it declared cyberinsecurity debt bankruptcy and built IIS 6.0 from scratch. That rewrite brought an untenably rising incidence of reported vulns down to a dull roar [5], as seen in Figure 1.

Of course, there are substantial security debts building elsewhere; here, in Figure 2 we demonstrate this buildup with some obvious choices, all on the same timeline as Figure 1, and their sum, which is a lower bound on net security debt buildup.

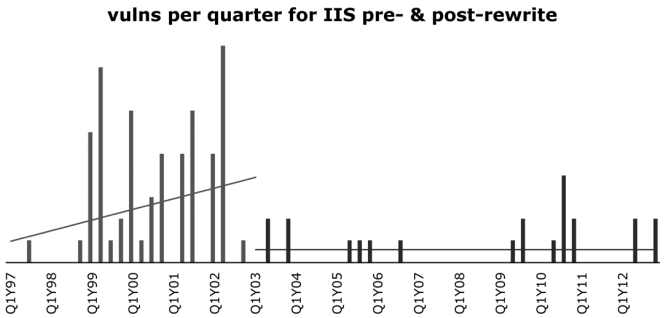


Figure 1: Rising incidence of reported vulnerabilities down

Financial bankruptcy is especially easy in the US, which is why the US economy rebounds from boneheaded financial mistakes faster than elsewhere—you just throw out the trash . . . unless the thing "you" need to bankrupt is too big to fail (TBTF). Now you can't shuck the debt. TBTF in finance is a bank whose failure would kill other firms. TBTF in cyber is an installed base too big to overwrite. As with Marsh Ray's TLS renegotiation attack [6], an installed base that is TBTF means that all that can be done is to add mitigating software on top of it. Adding software increases the attack surface. Happy New Day.

In the previous installment of this column [7], we proposed a market approach to dealing with cyberinsecurity risk by separating out severity from frequency of cyberinsecurity events. Severity is context dependent and a matter of taste. Frequency is an objective, mensurable fact, thus it can be the basis for a market. In synopsis, a futures market in the frequency of cyberinsecurity events (a trendline based on a counting function) dodges the question of severity (the maximum excursion of some unhappy cost curve). Trendlines are ordinal-scale statistics, i.e., good enough for decision support. Trendlines do not require the precision of definitions (what is "severity?") that frustrate the appearance of a hard science of cybersecurity. The key to the proposed market in cyberinsecurity event futures is an underlying debt pool from which the security events come, an underlying debt pool for which the security events provide an estimate. That underlying debt pool is, obviously, accumulated cyberinsecurity debt. A street cop cannot know how much heroin is for sale, but he can follow the price and adjust his policing based on which strategies raise the price of heroin. A cybersecurity cop cannot know how many vulnerabilities are present in the code on which he depends, but he can follow the price of cyberinsecurity event futures (and not the price of zero-days).

If cyberinsecurity insurance is written as a fixed dollar amount per cyberinsecurity event, then the predicted exposure of the insurer is simply the predicted frequency of cyberinsecurity events. And if cyberinsecurity events are, in turn, a linear function of cyberinsecurity debt load, then we have a third alternative (hedging in cyberinsecurity event futures) to what had been

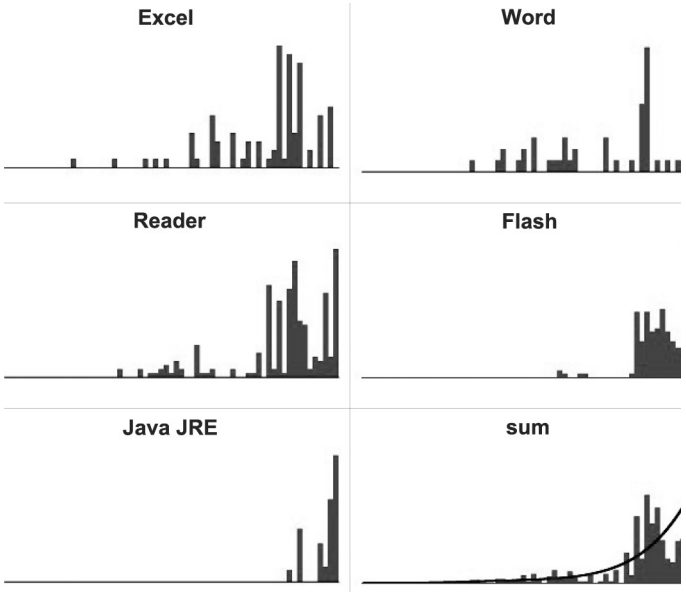


Figure 2: Security debts building

a choice of two less attractive alternatives: continuing to roll over the cyberinsecurity debt (of unknown size) or paying that debt down through codebase bankruptcy.

We consider Adobe's recent conversion to Software as a Service [8] to be an unacknowledged cyberinsecurity debt bankruptcy with Adobe remaining as a debtor in possession. Perhaps cyberinsecurity debt avoidance explains part of why the market capitalization of the top three SaaS vendors is growing five times as fast as the top three (product) sales vendors [9], as shown in Figure 3.

The collectivization of risk can be voluntary (you buy insurance) or involuntary (you are taxed to bail out TBTF). Insurance at industrial scale requires reinsurers—entities that sell insurance to insurers such as for linked-losses, viz., catastrophes where a single event (hurricane) causes large numbers of losses. The

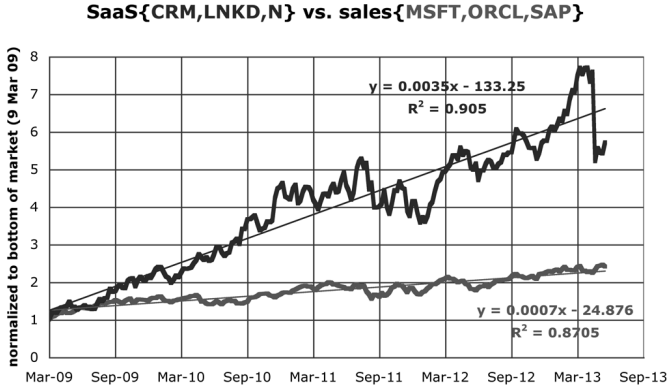
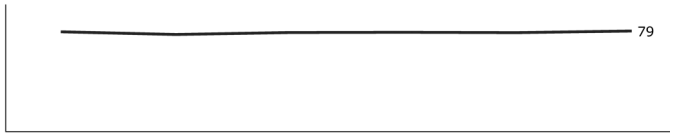


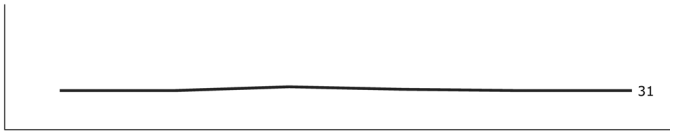
Figure 3: The market capitalization of the top three SaaS vendors is growing five times as fast as the top three sales vendors.

chance of catastrophes in cyberinsecurity is proportional to deployed interdependence, meaning that installed base is a strict lower bound indicator for what is TBTF in cyberinsecurity, i.e., the likelihood of a successful attack on one component of that interdependence may be a consequence of a successful attack on some different component.

Excepting TBTF, there always comes a point where risk transfer (like insurance) is a better investment than continued risk reduction, particularly when risk reduction is proven difficult even for firms that want to do it. Veracode's SoSS 5 report [10] shows two examples where among committed firms and repeated interventions, their cyberinsecurity score is all but constant. Over six quarters, analyzed software in the aggregate had a ±1.7% score



and Web applications subject to SQL injection had a ±4.7% score



Yes, those are flat lines.

Although it is true that the number of cyberinsecurity insurers is rising, so far as we know there is not yet a reinsurance market for cyberinsecurity insurance. Because reinsurance is a necessary condition for a robust market among primary insurers, and because the optimal number of reinsurers is the square root of the number of primary insurers [11], if the number of primary cyberinsecurity insurance issuers is to grow, so will grow the need for market makers in insurance futures. Despite being inconsistent with a free people, when a jurisdiction requires that its citizens buy insurance, capital must be sequestered to cover probable losses. A market that capitalizes the reserves needed for cyberinsecurity insurance is thus essential by observation: the risk is already collectivized even if merely ignored through the rolling over of cyberinsecurity debt society-wide.

One can argue about what is the "interest" on the cyberinsecurity debt, but it is unclear which of several models is relevant to the fundamental decisions—unless the interest rate is near zero, which it can only be by fiat rather than being market derived. The supply side makes exactly that assertion: the high-order bit on every page of every EULA is "It is not our fault," and courts have tended to agree that if the end user accepted such license terms, then they do govern. We do not think that cheaply rolling over cyberinsecurity debt can indefinitely continue, and therefore there needs to be a way to do risk transfer—one where objective measures of cyberinsecurity debt help price the transfer of risk. It would be wise to have that pricing in place before the rate shock hits.

References

[1] Ward Cunningham, "The WyCash Portfolio Management System," March 26, 1992: c2.com/doc/oopsla92.html.

[2] Chris Wysopal, "Application Security Debt and Application Interest Rates": http://www.veracode.com/blog/2011/02/application-security-debt-and-application-interest-rates.

[3] Elizabeth Nichols, "State of Software Security": http://www.veracode.com/blog/2013/05/soss-one-figure-at-a-time/.

[4] "Familiarity Breeds Contempt: The Honeymoon Effect and the Role of Legacy Code in Zero-Day Vulnerabilities": http://www.acsac.org/2010/openconf/modules/request.php?module=oc_program&action=view.php&a=&id=69&type=2.

[5] Data courtesy of osvdb.org.

[6] "Vulnerable Compliance": https://www.usenix.org/system/files/login/articles/geer.pdf.

[7] Dan Geer and Dan Conway, "The Price of Anything Is the Foregone Alternative,";login:, vol. 38, no. 3, June 2013: geer.tinho.net/login/geer.login.1306.pdf.

[8] "Adobe Kills Creative Suite—All Future Features Online Only": http://www.theregister.co.uk/2013/05/06/adobe_kills_creative_suite_for_cloud.

[9] Data courtesy of Yahoo Finance.

[10] "State of Software Security Volume 5," https://www.veracode.com/images/pdf/soss/state-of-software-security-report-volume5.pdf.

[11] Michael Powers and Martin Shubik, "A 'Square-Root Rule' for Reinsurance," *Revista de Contabilidade e Finanças* (Review of Accounting and Finance), vol. 17, no. 5, pp. 101-107.