DOI:10.1145/3746131

Carlos Barreto and Ulrik Franke

Opinion
Cyber Insurance, Software Diversity, and AI Coders

New automation possibilities offered by AI may change intrinsic properties of cyber risks.

Cyber insurance has been seen as a crucial tool in cybersecurity for two reasons: First, insurance against cyber incidents provides financial stability once an incident occurs, covering losses and offering services to recover systems (for example, investigation of incidents, legal defense, ransomware negotiations). Second, cyber insurance creates incentives for security by requiring security standards to get insurance at all, and by premium discounts to customers with better security.

However, cyber insurance has not grown as expected, and may now be at a turning point. The culprits are two characteristics of cyber risks: First, the difficulty to quantify rapidly evolving cyber risks based on historical data—needed for premiums to match actual risk. Second, the potential for incidents to spread between those insured, creating correlated losses and increasing the risk of multiple simultaneous insurance claims, known as accumulation risk.

In general, insurance assumes some independency between events, enabling portfolio risk management. Concretely, although the cost of a single event is uncertain, the average cost of multiple events can be estimated accurately. Thus, by pooling risks, an insurer can charge a fixed premium and have high certainty of being able to cover future claims. But as noted previously, some cyber risks are dependent. For instance, the NotPetya attack exploited security flaws in multiple computers and leveraged the networked nature of IT systems to propagate globally.[6]

Though risks can rarely be reduced to zero, sometimes there is an opportunity to mitigate their consequences.

Such accumulation risk was identified as a main barrier for insurance coverage a long time ago,[2] and it remains so today, in the age of ransomware. In the past, insurers have reacted to widespread cyberattacks by excluding losses resulting from state-sponsored cyberattacks. Disagreements interpreting policy exclusions have led to legal disputes, such as the litigation between Merck and insurance companies that declined coverage associated with the NotPetya malware—eventually settled in favor of the insured. Thus, if accumulation risks cannot be managed, cyber insurance may become too expensive or have too many exclusions to be meaningful or customers may be refused insurance altogether.

Natural catastrophes, epidemics, and terrorism also exhibit accumulation risk, but they can still be insured using the international reinsurance market. Although an earthquake may create many national claims simultaneously, internationally it may be an

IMAGE BY ANDRIJ BORYS ASSOCIATES WITH ASSISTANCE OF SHUTTERSTOCK.AI

isolated event—earthquakes do not propagate from Italy to California. Likewise, terrorism has been insured with national governments as reinsurers of last resort. Similar approaches are possible for cyber insurance as well.[1] However, the man-made nature of cyber incidents also opens another interesting avenue—changing the nature of the risks: in this *Communications* Opinion column, we explore this idea further.

Revisiting Software Diversity

Though risks can rarely be reduced to zero, sometimes there is an opportunity to mitigate their consequences. For example, construction standards can reduce losses from earthquakes and moderate use of alcohol and tobacco can reduce future health issues. However, natural disasters or genetic disorders prevent us from avoiding these risks completely. Similarly, cyber risks cannot be completely avoided, but we claim they could be controlled to a greater extent. In particular, some security practices that promote *software diversity* may change intrinsic properties of cyber risks; especially their correlation. (To be clear, software diversity is not diversity in the sense of DEI parlance. However, the fact that software diversity can improve resilience, at least in some circumstances, may be a valuable lesson also in other contexts.)

Software diversity is a longstanding strategy for fault-tolerance that creates redundancy in multiple components that have identical external behavior but different internal mechanics. This reduces the probability of having identical faults in two versions of a component. This is crucial for cybersecurity, as it can prevent the reuse of attacks and limit the spread of malware that exploits particular vulnerabilities, just as more biodiverse populations are more resilient to biological viruses.[8] Concretely, diverse systems can be more resilient to various attacks and also take longer to compromise.[7]

Diverse systems can be more resilient to various attacks and also take longer to compromise.

Some software diversity is natural as computing systems are often built with different operating systems, Web servers, and database management systems. Indeed, continuously updated libraries may make it very difficult to compile identical versions of programs at different points in time. However, deliberate effort is needed to reach significant protection. Thus, the literature has focused on how to intentionally introduce software diversity, and on managing the resulting systems.[10] The most well-known software diversity mechanism is probably moving target defense (MTD), a family of mechanisms that gained fame through the development and deployment of address space randomization in popular operating systems approximately 20 years ago.

The essence of moving-target defense is to take advantage of the asymmetric knowledge of attackers and defenders.[3] This is done by introducing randomness, such as changing the network structure or the low-level execution of a program. Such changes to the attack surface can improve security, first, because attacks may take longer (changes in the system can create setbacks in the progression of the attack), and second, because randomness in execution or operation can prevent the reusability of attacks.

While there is much academic work on software diversity, there is reason to believe it is not yet employed to its full potential in practice. To be sure, some applications, such as address space randomization, are widely employed and others like *N*-version programming—where *N* independent implementations must agree to make critical decisions—are used in niches. But there has not been enough research on software diversity metrics, on the interplay between hardware and software diversity, on the theoretical underpinnings of software diversity, or on how it relates to other dependability and security attributes.[10] In practice, the difficulties and costs of managing truly diverse systems—maintaining parallel setups of various hardware, operating systems, languages, compilers, and external libraries, and pushing all of this through automated testing regimes, deployment processes and user support—have often not seemed worth the benefits. But this need not necessarily hold true in the future.

Security mechanisms often aim for more security in a technical sense—to decrease the number of successful attacks. It is easy to jump to the conclusion that if the number of successful attacks is *not* decreased, the security mechanism is a failure. But this is premature! Even if, pessimistically, software diversity mechanisms make systems just as good or bad as systems with less software diversity *on average*, software diversity mechanisms might introduce the randomness needed to decrease the correlation of incidents. This would be the equivalent of turning a global natural disaster into several smaller, local events. From a myopic perspective this may look like a questionable gain, but crucially, this may be enough to turn uninsurable systems into insurable ones. Adopting a wider perspective, this may be a considerable gain for society-at-large, including those affected, who would not need to stockpile cash to self-insure (or go bankrupt if they do not) but can instead pay affordable insurance premiums. Enabling insurance markets is a boon—hence the concern and worry about cyber insurance being increasingly unavailable.

Taking this possibility of insurance into account, reducing the *correlation* between events can be more important than reducing the *impact* of individual incidents. Making risks less heavy-tailed may also make them easier to estimate,[4] entailing more accurate premiums (current premiums are not actuarial[5]). This suggests that more software diversity could be a game-changer for cyber insurance. Optimistically, requiring software diversity to get insurance could trigger a cascade effect reaching multiple industries, provided that multiple insurers or other players with considerable market power (for example, a reinsurer or a government) adopt such policies. More pessimistically, firms may try to game the system and adopt software diversity only to qualify for insurance coverage, neglecting other security controls. The result of the pessimistic scenario would be an ecosystem with diverse but insecure systems. Though hardly optimal, even this may nevertheless be preferable to the current ecosystem of non-diverse and insecure systems—the silver lining being that premiums

While there is much academic work on software diversity, there is reason to believe it is not yet employed to its full potential in practice.

in the more diverse ecosystems would better reflect real risks, thus eventually providing that argument needed to force boards to invest in security, long coveted by CISOs.

Software Diversity at a Discount?

As previously mentioned, software diversity may have struggled in practice, because of quality and reliability concerns and because of perceived costs. Development, testing, deployment, patching, and support becomes more difficult. Furthermore, diversification interventions can degrade performance, such as execution time. Indeed, the randomness, which is the hallmark of software diversity, may from another perspective be the definition of poor quality.

Nevertheless, one reason to revisit software diversity and MTD today is that advances in AI, applied to code generation, can reduce the cost of software diversity. At the moment, there is a flurry of academic and practitioner interest in how AI-assisted code-generation tools, such as GitHub Copilot, Amazon CodeWhisperer, and ChatGPT can boost productivity. Of course, the code generated by such tools is not free from errors, and it may be that the greatest productivity boost comes not from replacing human software engineers but from finding the best way to combine human and artificial intelligence. But while the general problem of AI code generation and how to best use it in organizations is multi-faceted, our concern here is first and foremost code diversity, and as it turns out, some such diversity is very straightforward to achieve with AI code-generation methods. For example, AlphaCode generates many potential solutions for a single problem, with different execution times and memory usage, and then goes on to filter out a small set of—probably—correct programs.[9] Thus, if an n-fold increase in software diversity 10 or 20 years ago may have required hiring (almost) n times more coders, this is no longer the case. Automated code generation cannot make software diversity measures free—especially not when the complexity of testing, deployment, patching, and so forth is taken into consideration—but it can make some versions of it much cheaper.

Cyber insurance offers important potential benefits but has been a disappointment because of correlated losses.

A Confluence Requiring Governance

Cyber insurance offers important potential benefits, but has been a disappointment because of correlated losses. Correlated losses can be addressed using software diversity, but such measures have failed to succeed in practice because they are too cumbersome and expensive. Now, advances in automated code generation could make software diversity much cheaper. This leads to the question: Is automated code generation now cheap and reliable enough to enable enough software diversity to salvage cyber insurance? While we believe this is a very pertinent question to ask, the answer is not obviously positive. The costs of software diversity mechanisms—though smaller—may still be too large. AI-driven software diversity measures might face resistance within development teams or organizations that may be hesitant to trust automated processes for critical code. While in theory, insurers should be pushing for software diversity to disassemble accumulation risk, their market power and technical know-how may be too small. Intelligent attackers may circumvent the software diversity introduced.

Nevertheless, we believe there is a very interesting confluence here, one that merits attention from practitioners and academics alike. Start with the research questions. Many previously identified knowledge gaps remain, for example, proper metrics, the interplay between hardware and software diversity, and how it relates to other dependability attributes.[10] But the cyber insurance context entails a new important perspective: How should software diversity look if it is enough to bring down the *variance* rather than the *mean* of the damage caused by cyber threats?

Turning to practitioners, the big question is one of governance. If enterprises in some particular sector—private or public—are currently denied cyber insurance, can they introduce software diversity to make themselves insurable? Individual enterprises may struggle to do so, but collectively, it is not impossible to introduce new standards or put pressure on vendors for software diversity. To do so, however, they may need insurers and possibly also regulators onboard from the very beginning, to make the business case. Ultimately, a balance must be struck between the benefits and challenges. Software diversity may not be a panacea for everyone to embrace. But if introduced in the right places, it may break the economies of scale that make cyber-crime so lucrative. As such, it is certainly worth exploring. C

References

1. Barreto, C., Schwartz, G., and Cardenas, A.A. Cyber-Insurance. *Safety, Security and Privacy for Cyber-Physical Systems*. Springer International Publishing (2021).
2. Böhme, R. and Kataria, G. Models and measures for correlation in cyber-insurance. *Workshop on Economics of Information Security* (2006).
3. Carvalho, M. and Ford, R. Moving-target defenses for computer networks. *IEEE Security and Privacy 12*, 2 (2014).
4. Edwards, B., Hofmeyr, S., and Forrest, S. Hype and heavy tails: A closer look at data breaches. *J. Cybersecurity 2*, 1 (2016).
5. Franke, U. The cyber insurance market in Sweden. *Computers and Security 68* (2017).
6. Greenberg, A. The untold story of NotPetya, the most devastating cyberattack in history. *Wired* (Aug. 22, 2018).
7. Hole, K.J. Diversity reduces the impact of malware. *IEEE Security & Privacy 13*, 3 (2013).
8. Knight, J.C. *Diversity*. Springer, Berlin Heidelberg (2011).
9. Li, Y. et al. Competition-level code generation with AlphaCode. *Science 378*, 6624 (2022).
10. Zhang, Q. et al. Diversity-by-design for dependable and secure cyber-physical systems: A survey. *IEEE Trans. on Network and Service Management 19*, 1 (2021).

Carlos Barreto (carlobar@gmail.com) was previously a postdoctoral researcher with the Digital Futures program at KTH Royal Institute of Technology in Stockholm, Sweden.

Ulrik Franke (ulrik.franke@fhs.se) is a senior lecturer at the Swedish Defence University, Stockholm, Sweden, and was previously an adjunct professor at KTH Royal Institute of Technology.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/cyber-insurance