Formal Methods and CyberSecurity

James Davenport

University of Bath
Former Fulbright CyberSecurity Scholar

4 September 2019

CyberSecurity

- CyberSecurity failures abound: tens daily in the specialist press, and every few weeks as mainstream news

! More frequently than train crashes, and much more than aeroplane crashes

- Many people affected: 148 million for Equifax [Blo18] and probably more for the Starwood breach: [BBC18] states 500 million
- The financial costs can be substantial: bankruptcy in the case of American Medical Collection Agency [For19] and a provisional £183M fine for British Airways [The19]
- There are many reasons for CyberSecurity failures, and a given failure may have many: [Uni18] " identified four major factors including identification, detection, segmenting of access to databases, and data governance that allowed the attacker . . ."

Fundamentally, there was a bug [Len17] of a well-known kind, easy to flag automatically

Formal Methods: a range of ideas and tools

- At one end, the use of a theorem-prover and associated tools to prove formal statements about a program ("not crashing", "not deadlocking", "maintaining certain invariants") etc.
- Similar, but proving certain faults ("buffer overflow" etc.) can't occur
- Or proving statements about information flow ("taint analysis")
- Down to simple syntactic tools

The safety-critical industry (trains, aeroplanes etc.) would not dream of doing without these tools, and generally insists on the formal proof of key properties

But in CyberSecurity . . .

The Payment Card Industry [Pay18] has two relevant requirements.

6.5 Address common coding vulnerabilities in software-development processes as follows:
- Train developers at least annually in up-to-date secure coding techniques, including how to avoid common coding vulnerabilities;
- Develop applications based on secure coding guidelines.

6.6 For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by either:
- Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes;
- Installing an automated technical solution that detects and prevents web-based attacks in front of public-facing web applications, to continually check all traffic.

Essentially, 6.6 admits that 6.5 isn't sufficient, and 6.5 has no tool/methodology requirement

Actually, one can ask if they do as well!

Equifax bought a product (Apache) with no such guarantees, and didn't check or update it

6.6 "prevents web-based attacks" is Turing-complete, so we have is "prevents known web-based attacks"

50% "of security breaches are caused by coding errors" [McG06]

Forever 21 breach caused by disregard of PCI requirements [Pay18] — [Bis18], also Macy's [Bla18]

Ticketmaster Failure to communicate requirements [Inb18]

Of course, there are successes

1. Using technology (SPARK Ada subset) from the safety-critical industry, there is a secure download system for embedded systems [Cha18]

... Can anyone name another one?

It is depressing that what is billed as "set up a trustworthy, self-improving and resilient digital environment that can thrive in the face of unanticipated threats, and earn the trust people place in it" [Roy16] has only one mention of formal methods: "The application of formal methods to safety critical applications".

Maybe our goals are too high?

Some major developers are moving

and placing confidence in the use of verification tools *as well as* conventional testing.

- **AWS** (Amazon Web Services) [Vog19]: "Zelkova does this [alerting customers] by using automated reasoning to analyze policies"; "Tiros maps the connections between network mechanisms".
- **+** Very interesting, proof about configurations
- **Google** [SAE+18]: "Many of the static analysis tools deployed at the scale of Googles two-billion-line codebase are relatively simple"
- **Facebook** [DFLO19] "Infer targets our mobile apps as well as our backend C++ code"; "Zoncolan targets the 100-million lines of Hack code"

Why is CyberSecurity different?

The answer perhaps lies in the fact the security is seen, even by developers, as an optional extra [TV19]

- "security is not currently seen as part of working software, it only costs extra time and it doesn't provide functionality" [vdHBS18]

This is most evident in the "Agile" mindset: attackers don't write user stories.

Is the education process partly to blame [CDIP19]?

! Teachers rarely have the time to do the detailed code reviews that would reveal security problems (where relevant)

? And is the ratio of programming assignments that involve security at all like the real-life ratio?

So the users are different

Or, at least, more sensitive

Google "Our most important insight is that careful developer workflow integration is key for static analysis tool adoption" [SAE+18]

Facebook Switching Infer from batch mode to operating at diff time moved the fix rate from 0% to 70%

+ essentially by avoiding a context switch

! in the programmer's brain

This is known in safety-critical contexts: [BS12] shows how incremental verification can take "time for a coffee", rather than overnight, and this is key to productivity

The scale is certainly different

Safety [BS12] had programs from 100k–1M lines.

results in less than 5 minutes

Google 2G lines of code.

Google does not have infrastructure support to run interprocedural or whole-program analysis at Google scale.

Facebook "over 100M lines of Hack code, which Zoncolan can process in less than 30 minutes." "We have 10s of millions of both mobile code and backend C++ code"

"Infer processes the code modifications quickly (average 15 minutes)"

Conclusions

- There is room for even "trivial" tools to improve security code
- The scale issues are challenging, but recent progress is very encouraging
- For a variety of reasons, current programming languages are not well-suited to accuracy: Google's "Zero Day" project reports [Goo19] that 68% of zero-day exploits were caused by memory corruption errors, and Microsoft report a very similar story [Tho19].

Many web pages are JavaScript, with very non-local semantics, and much inclusion of third-party code [ZML+19], which leads to many attacks.

?? Should the CyberSecurity industry be starting from here?

BBC.
Marriott hack hits 500 million Starwood guests.
https://www.bbc.co.uk/news/technology-46401890, 2018.

C. Biscoe.
MyFitnessPal data breach: 150 million app users affected.
https://www.itgovernance.co.uk/blog/
myfitnesspal-data-breach-150-million-app-users-affecte
2018.

A. Blackmon.
Macy's hit by data breach.
https://eu.freep.com/story/money/business/2018/07/06/macys-data-breach-online/763074002/, 2018.

Bloomberg.
Equifax Hack Lasted for 76 Days, Compromised 148 Million People, Government Report Says.
http://fortune.com/2018/12/10/equifax-hack-lasted-for-76-days-compromised-148-millio
2018.

M. Brain and F. Schanda.
A lightweight technique for distributed and incremental verification.
In Rajeev Joshi, Peter Müller, and Andreas Podelski, editors, *Verified Software: Theories, Tools, Experiments*, volume 7152 of *LNCS*, pages 114–129, Berlin–Heidelberg–New York, January 2012. Springer.

T. Crick, J.H. Davenport, A. Irons, and T. Prickett.
A UK Case Study on Cybersecurity Education and Accreditation.
https://arxiv.org/abs/1906.09584, 2019.

R. Chapman.
Development and Formal Verification of Secure Updates for Embedded Systems (slides from Verification 2018).
http://www.testandverification.com/conferences/verification-futures/vf2018/, 2018.

D. Distefano, M. Fähndrich, F. Logozzo, and P.W. O'Hearn.
Scaling static analyses at Facebook.
Communications of the ACM, 62(8):62–70, 2019.

N. Ford.
Medical debt collection agency files for bankruptcy protection after data breach.
https://www.itgovernance.co.uk/blog/medical-debt-collection-agency-files-for-bankruptcy-pro
2019.

Google (Project Zero).
0day "In the Wild".
https://googleprojectzero.blogspot.com/p/0day.html, 2019.

Inbenta (CEO).
Inbenta and the Ticketmaster Data Breach.
http://web.archive.org/web/20181121184620/, 2018.

L. Lenart.
Security Bulletin S2-045.
https://cwiki.apache.org/confluence/display/WW/S2-045, 2017.

G. McGraw.
Software Security — Building Security In.
Addison-Wesley, 2006.

Payment Card Industry Security Standards Council (PCI SSC).
Requirements and Security Assessment Procedures Version 3.2.1.
https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf, 2018.

Royal Society.
Progress and research in cybersecurity: Supporting a resilient and trustworthy system for the UK.
http://royalsociety.org/cybersecurity, 2016.

C. Sadowski, E. Aftandilian, A. Eagle, L. Miller-Cushion, and C. Jaspan.
Lessons from building static analysis tools at Google.
Commun. ACM, 61(4):58–66, 2018.

The Guardian.
BA faces £183m fine over passenger data breach.
https://www.theguardian.com/business/2019/jul/08/ba-fine-customer-data-breach-british-airways, 2019.

G. Thomas.
A proactive approach to more secure code.
https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/, 2019.

M. Tahaei and K. Vaniea.
A Survey on Developer-Centred Security.
https://groups.inf.ed.ac.uk/tulips/papers/A_Survey_on_Developer_Centred_Security.pdf, 2019.

United States Government Accountability Office.
Actions Taken by Equifax and Federal Agencies in Response to the 2017 Breach.
https://www.gao.gov/assets/700/694158.pdf, 2018.

A. van der Heijden, C. Broasca, and A. Serebrenik.
An empirical perspective on security challenges in large-scale agile software development.
In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, pages 45:1–45:4, New York, NY, USA, 2018. ACM.

W. Vogels.
Proving security at scale with automated reasoning.
https://www.allthingsdistributed.com/2019/05/proving-security-at-scale-with-automated-reasoning.html, 2019.

M. Zhang, W. Meng, S. Lee, B. Lee, and X. Xing.
All Your Clicks Belong to Me: Investigating Click Interception on the Web.
https://www.microsoft.com/en-us/research/uploads/prod/2019/03/zhang-observer.pdf, 2019.