Issue 087: Considered Harmful

De Programmatica Ipsum

2025-12-01

Contents

Issue 087: Considered Harmful

By Adrian Kosmaczewski, December 1st, 2025

Welcome to the 87th issue of *De Programmatica Ipsum*, about *Considered Harmful*.

In this edition:

- We weigh the historical and philosophical implications of the phrase "Considered Harmful"[1].

[1] https://deprogrammaticaipsum.com/harmfully-considered/

- In our Vidéothèque section[2], we watch Phil Nash[3] explain which parts of OOP can be considered harmful.
- In the Library section[4], we read a paper by Joanna Rutkowska[5] explaining why the Intel x86 architecture should be considered harmful.

Download this issue in DRM-free PDF[6] or EPUB[7] format, and read it on your preferred device. You can also subscribe to our RSS feed[8], featuring the full content of our articles. We also invite you to check the new "Security" section[9], with a selection of articles devoted to the subject.

We would like to thank our patrons who generously contribute every month (or have contributed in the past) to our work and help us run this magazine. Thank you so much! In alphabetical order: Adam Guest, Adrian Tineo Cabello, Benjamin Sheldon, Christopher Nascone, Colin Powell, Franz Lucien Moersdorf, Guillermo Ramos Álvarez, Jean-Paul de Vooght, Dr. Juande Santander-Vela, Patryk Matuszewski, Paul Hudson, Quico Moya, Roger Turner, Szymon Licau, and countless more leaving anonymous tips every month.

Enjoy this issue! Please share our articles on social media, or contribute[10] if you would like to support our work with a donation via Liberapay[11].

Cover photo by Егор Камелев[12] on Unsplash[13].

[2] https://deprogrammaticaipsum.com/category/videotheque/
[3] https://deprogrammaticaipsum.com/phil-nash/
[4] https://deprogrammaticaipsum.com/category/library/
[5] https://deprogrammaticaipsum.com/joanna-rutkowska/
[6] https://deprogrammaticaipsum.com/pdf/issue-087-considered-harmful.pdf
[7] https://deprogrammaticaipsum.com/epub/issue-087-considered-harmful.epub
[8] https://deprogrammaticaipsum.com/index.xml
[9] https://deprogrammaticaipsum.com/category/security/
[10] https://deprogrammaticaipsum.com/contribute/
[11] https://liberapay.com/akosma/donate
[12] https://unsplash.com/@ekamelev?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[13] https://unsplash.com/photos/red-mushroom-selective-focus-photography-aJCiOy846G8?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Harmfully Considered

By Adrian Kosmaczewski, December 1st, 2025

We, the authors of the magazine you are reading right now, sometimes joke that the true legacy of these articles would be realized if someone presented a paper titled “De Programmatica Ipsum Considered Harmful”. Alas (or not), we do not think this is ever going to happen, although I know that some members of academia follow our ramblings month after month, so who knows; if you ever publish such a contraption, we would love to print and frame a copy for the sake of bragging rights.

Speaking about jokes, "Considered Harmful" is one of the most recurring ones in the history of programming. Not that the countless papers, blog posts, videos, podcast episodes, and conference talks that featured these two words as a suffix were funny, but because it is a trigger: one that fires up specific alarms in the systems of software developers, immediately commanding our utmost attention.

Again, for good or for worse.

In his excellent 2012 book "The Dawn of Software Engineering: From Turing to Dijkstra", one that truthfully deserves its own article on the Library section of this magazine, Edgar G. Daylight provides a surprising and well-researched chronicle of the birth of Structured Programming[14] (with leading capitals, thankyousomuch): a now legendary paper[15] that, if you have not read it, you must most probably know about thanks to previous articles[16] of this magazine.

We thus read between pages 33 and 35 of Daylight's opus that

> In 1962, after his IFIP address in Munich, Dijkstra became Professor of Mathematics in Eindhoven and started working on the THE operating system (...) It is in this setting that Dijkstra conceived of the idea of writing his 1968 letter 'Go To Statement Considered Harmful'. (...)
>
> At the end of his letter, Dijkstra acknowledged that Zemanek had expressed doubts about the goto statement as early as 1959, and also mentioned Landin, Strachey, and Hoare as inspirators. (...)
>
> Nevertheless, it was Dijkstra's letter which created the most waves. Besides attracting immediate attention, Dijkstra's letter polarized the computer community. (...)
>
> Dijkstra said he received "a torrent of abusive letters" after his letter was published. In subsequent years, articles titled "X considered harmful" were published with in one case even "Dijkstra considered harmful". (...)

(Kids, this was decades before social media.)

Daylight, however, blows the whistle and points to the true culprit of the flame wars

[14] https://deprogrammaticaipsum.com/edward-nash-yourdon/

[15] https://dl.acm.org/doi/10.1145/362929.362947

[16] https://deprogrammaticaipsum.com/programming-the-liberal-arts/#:~:text=Take%20that,%20Edsger

that ensued, and you would never believe who that was.

> Dijkstra's letter was not in accord with his original intentions, however. Initially, Dijkstra had submitted a paper to the editor of the Communications of the ACM, Niklaus Wirth, under the moderate title 'A case against the goto statement'. It was the editor who, in the interest of speeding up publication, single handedly decided to publish the paper as a letter under the far more provocative title 'Go To Statement Considered Harmful'.

Ahahah! So it was our most dear Mr. Wirth[17] who was behind this snowclone[18] all the time.

Two decades later, Frank Rubin started a long and heated debate[19], held over reader letters published on the Communications of the ACM magazine. The starting point of the debate was a letter titled " 'GOTO Considered Harmful' Considered Harmful", a self-referential title akin to the meaning of the GNU acronym.

> It is like butchers banning knives because workers sometimes cut themselves. Programmers must devise elaborate workarounds, use extra flags, nest statements excessively, or use gratuitous subroutines. The result is that GOTO-less programs are harder and costlier to create, test, and modify.

A few months later, Dijkstra lamented[20] the tone and content of the debate.

> The whole correspondence was carried out at a level that vividly reminded me of the intellectual climate of twenty years ago, as if stagnation were the major characteristic of the computing profession, and that was a disappointment.

The debate continued during the 1990s and 2000s. Ward Cunningham's original "WikiWikiWeb" features a page about it[21]. Jeff Atwood[22] and Mark Liberman[23] poured virtual ink about the subject. There is a Wikipedia page titled "Considered harmful"[24], filled with crunchy details and references.

---

[17] https://deprogrammaticaipsum.com/niklaus-wirth/
[18] https://en.wikipedia.org/wiki/Snowclone
[19] https://www2.cs.arizona.edu/classes/cs372/spring17/gotoletters.pdf
[20] https://www.cs.utexas.edu/~EWD/ewd10xx/EWD1009.PDF
[21] https://wiki.c2.com/?ConsideredHarmful
[22] https://blog.codinghorror.com/id-consider-that-harmful-too/
[23] http://itre.cis.upenn.edu/~myl/languagelog/archives/004675.html
[24] https://en.wikipedia.org/wiki/Considered_harmful

Last but not least, Eric Meyer[25] wrote (in 2002!) an essay called "Considered Harmful" Essays Considered Harmful[26], which summarizes a lot of our thoughts about the subject, pointing out that

> The controversy resulting from the article's publication became so heated that the CACM subsequently decided to never again publish pieces with such assertive positions.

Ouch.

> "Considered harmful" essays are not only a sad cliché at this stage of the game, they are counter-productive to reasoned debate and most often do far more harm than good to whatever cause they promote.

2025 is about to end as I write these words, that is, 23 years after Eric Meyer wrote this, and well, clearly nobody paid attention. Content whose titles end with the words "Considered Harmful" keep on being featured on Hacker News and Reddit every so often.

In a more sarcastic tone, Pastor Manul Laphroaig[27] claimed in a sermon[28] that:

> Pouring myself another, I say "Dijkstra's advice goes well enough if you wish to program software. It is true that BASIC is a horrid language for writing complex software, but consider again the educational value of spaghetti code.
>
> "Dijkstra says that a mind exposed to BASIC can never become a good programmer. While I trust his opinion on algorithms, his thoughts on BASIC are racist horse shit.

Amen.

There are, however, two things we would like to say about "Considered Harmful". Spoiler alert: neither of them is positive.

First, there is an inherently abhorrent philosophical aspect that permeates the concept of "X Considered Harmful" in the minds of developers (in particular, younger generations thereof). By cataloging subject X as dangerous, it institutionalizes a rhetorical

[25] https://en.wikipedia.org/wiki/Eric_A._Meyer
[26] https://meyerweb.com/eric/comment/chech.html
[27] https://deprogrammaticaipsum.com/pastor-manul-laphroaig/
[28] https://archive.org/details/Pocorgtfo01/page/n13/mode/2up

violence around it, automatically signaling a tribal feeling of belonging.

Acknowledging the (supposed) evil described in a "Considered Harmful" piece, automatically place consumers as part of an enlightened group, all wearing the same imaginary rallying symbol around their necks, perpetuating a (potentially toxic) shibboleth[29] across time and space. Taken without context, such "ethical imperatives" imply moral and/or professional neglect from practitioners. Harassment, exclusion, and distress ensues.

Second, we think of it as a shortcut to intellectual laziness; "Considered Harmful" stuff is often used to shut down debate, to force an "objective" truth upon the minds of others, and to avoid all nuance in favor of a binary narrative.

The death of "it depends", if you will. The altar where contextual awareness is sacrificed.

Being human means, to a large degree, spending our younger years mindlessly identifying the enemy of our friend as our own enemy. On the other hand, growth means learning to spot the occasions when such thought patterns arise, and consciously elaborating on them, leaving dogma and hormones aside.

Software developers and programmers pride themselves (and rightfully so!) of the cognitive abilities they possess, allowing them to solve the most intricate problems with code. It is the fervent wish of the authors of this magazine, and a common thread across all of our writing, to see all of us use some of that CPU power to identify our own shortcomings and biases.

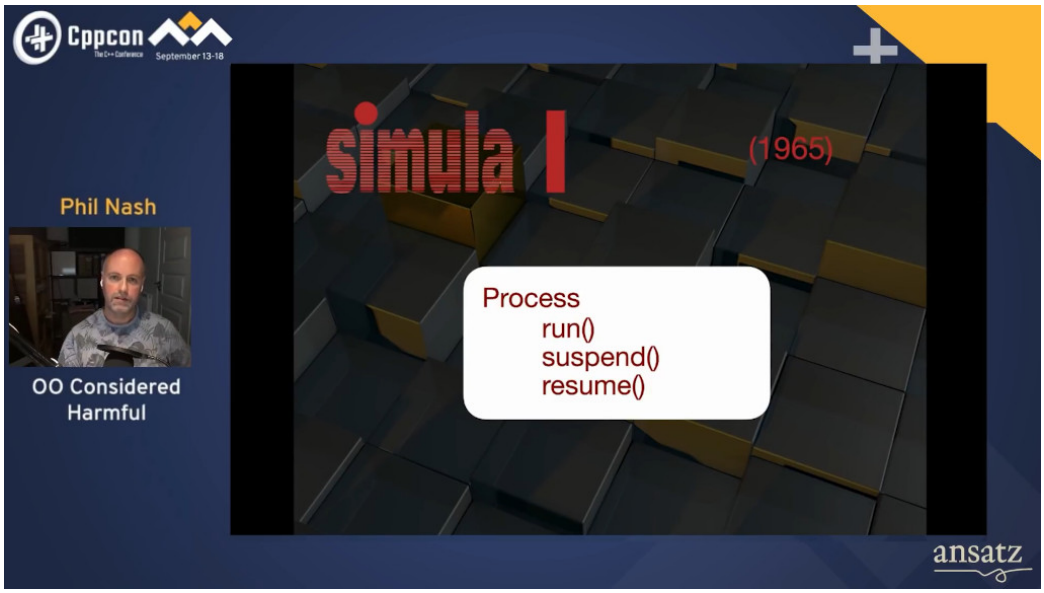
Cover photo by Bernd □ Dittrich[30] on Unsplash[31].

[29]https://en.wikipedia.org/wiki/Shibboleth

[30]https://unsplash.com/@hdbernd?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[31]https://unsplash.com/photos/black-and-white-no-smoking-sign-fq_LeogCtKM?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Phil Nash



By Adrian Kosmaczewski, December 1st, 2025

A quick search on YouTube with the query "Considered Harmful"[32] is a revealing exercise. The number and variety of articles thereby returned is outstanding and, to a certain extent, hilarious. The day I wrote this article I had the following ones popping up, all of which were literally considered harmful for the purposes of the content: threads[33],

[32] https://www.youtube.com/results?search_query=considered+harmful
[33] https://www.youtube.com/watch?v=_T1XjxXNSCs

enums[34], C++ generics[35], `rand()`[36], `if`[37], `else`[38], the `UPDATE`[39] SQL statement, global variables[40], user stories[41], architecture[42], YAML[43] (well, this one we can agree upon), `IInterface`[44], mocking frameworks[45], assemblers[46], abstractions[47], penetration testing[48], and yes, even programming itself[49]. Among those search results there was even a talk by Alan Kay himself named "Normal Considered Harmful"[50].

(Small pause while I catch my breath.)

Have we reached the point where the phrase "Considered Harmful" itself has lost all meaning, or, even worse, it ends up signaling content of dubious quality or intentions? Maybe; there is still a part of me that tries to consume content with an open eye regardless of the title creators chose, and among all the chaff, there thankfully is some grain waiting to be found.

For this month's Vidéothèque issue, we have chosen Phil Nash[51]'s excellent talk titled "OO Considered Harmful"[52], delivered at CppCon 2020 (remotely, of course, as it used to be the case during the last pandemic).

Phil is a programmer[53], trainer, and a regular speaker in the C++[54] galaxy, and with reason: he is the original author the Catch2[55] unit test framework, highly regarded as one of the standard unit testing frameworks for C++. He is also co-host of the Cpp-

[34]https://www.youtube.com/watch?v=jjMbPt_H3RQ
[35]https://www.youtube.com/watch?v=jXQ6WtYmfZw
[36]https://www.youtube.com/watch?v=LDPMpc-ENqY
[37]https://www.youtube.com/watch?v=z43bmaMwagI
[38]https://www.youtube.com/watch?v=XDqZemwVXUk
[39]https://www.youtube.com/watch?v=JxMz-tyicgo
[40]https://www.youtube.com/watch?v=qrjLXkUtmY4
[41]https://www.youtube.com/watch?v=2df8_PHJuzk
[42]https://www.youtube.com/watch?v=l5Fbnvdj8xs
[43]https://www.youtube.com/watch?v=WQurEEfSf8M
[44]https://www.youtube.com/watch?v=uQV_jU05lFQ
[45]https://www.youtube.com/watch?v=uhuHZXTRfH4
[46]https://www.youtube.com/watch?v=rf6oFSEdbKI
[47]https://www.youtube.com/watch?v=tqikjhLOw6s
[48]https://www.youtube.com/watch?v=GvX52HPAfBk
[49]https://www.youtube.com/watch?v=VpuVDfSXs-g
[50]https://www.youtube.com/watch?v=FvmTSpJU-Xc
[51]https://www.linkedin.com/in/philsquared/
[52]https://www.youtube.com/watch?v=pH-q2m5sb04
[53]https://github.com/philsquared
[54]https://deprogrammaticaipsum.com/issue/issue-077-c++/
[55]https://github.com/catchorg/Catch2

Cast[56] podcast, and has worked as developer advocate for various companies including JetBrains and SonarSource, delivering talks[57] about programming and C++ for more than 20 years at the time of this writing.

In this magazine we are fervent fans of object-orientation[58]; so it should not be a surprise that this talk would catch our attention (no pun intended, Phil).

Phil starts his video by explaining the connection to Dijkstra's original paper, by exhorting developers to shorten the conceptual gap between the static code of a program and its dynamic representation in the computer, usually referred to as a process.

In this context, Dijkstra's critique of the `goto` statement is akin to that of mutable state; the former complicating control flow, the latter complicating data flow. Mutable state is a burden that lengthens the debugging phase for any program, and hence, its complicatedness.

Phil then takes a historical perspective; another favorite angle of this magazine. And he does this through the examples of languages like Simula and Smalltalk, Objective-C and Eiffel, all pioneering the object-orientation paradigm among programmers. I bet you have never seen a snippet of Simula in your life; well, here is your chance.

And then he gets into the interesting part: the critique of core OOP concepts, like polymorphism, inheritance, and encapsulation, all within the context of modern C++.

Regarding polymorphism, Phil states that the standard approach ("vtables") is usually avoided in performance-critical C++ code, and he proposes the use of type erasure (similar to Rust traits or Swift protocols) as a much better, non-intrusive, and inheritance-free alternative.

Regarding inheritance, he mentions something Rust and Go developers have noticed all along: you just do not need it, and mixins and traits are much more modern and flexible alternatives.

Finally, regarding encapsulation, he rightfully claims it to be the most valuable aspect of OOP, even though the implementation provided by C++ can be leaky sometimes (`protected` and `friend`, someone?) and does not reach the perfection of other languages like Erlang.

---

[56] https://cppcast.com/

[57] https://levelofindirection.com/appearances.html

[58] https://deprogrammaticaipsum.com/the-hype-cycle-of-oop/

So, what is the road ahead? According to Phil, "Value-Oriented Design"; a multi-paradigm approach that marries OOP with Functional programming ideas, leading to stronger, safer code. And how is this achieved? Through two basic mechanisms. First, immutable value types, created through the Builder design pattern, and managed through persistent data structures, allowing modifications that avoid copying, for performance gains. And second, by encapsulating "components", namely objects with internal state and processes, in such a way that they use clear interfaces to talk to one another.

In conclusion, the "harmful" part of OOP has to do with the use of shared mutable state and rigid inheritance hierarchies. Instead, build your software with immutable value types at the bottom, and encapsulated components on top of them.

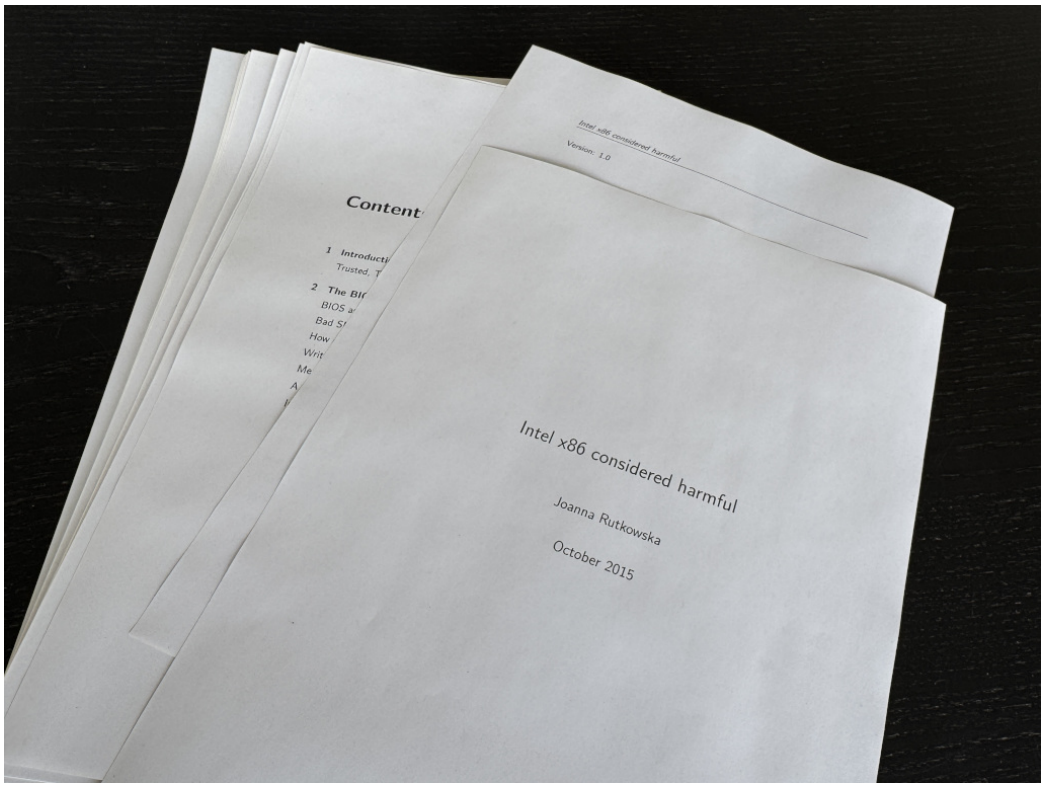
So maybe, just maybe, the future of programming (besides the reliance on AI-based coding tools) is not just adhering to a single paradigm, but rather combining the strengths of many to make code that is safe and performant at the same time. Once again, we sense that idea floating in the air, that the most "harmful" thing we can do as programmers is to blindly swallow dogmas[59], not considering the various options that we have at our disposal, and how to make them work in harmony.

Watch this month's Vidéothèque entry, "OO Considered Harmful"[60] by Phil Nash, on YouTube.

Cover snapshot chosen by the author.

[59] https://deprogrammaticaipsum.com/you-are-doing-it-wrong/
[60] https://www.youtube.com/watch?v=pH-q2m5sb04

Joanna Rutkowska



By Adrian Kosmaczewski, December 1st, 2025

In a key scene of the 2012 blockbuster James Bond film "Skyfall"[61], MI6 quartermaster Q, played by Ben Whishaw, realizes too late[62] that plugging a cable into the laptop of

[61] https://en.wikipedia.org/wiki/Skyfall
[62] https://www.youtube.com/watch?v=aApTVqeGJMw

a notoriously skilled terrorist like Raoul Silva (one of Javier Bardem's most remarkable roles) was a terrible idea. After a few seconds of connection, the laptop infects the systems of MI6, releasing all physical doors and disabling all security guards, prompting Silva to escape and wreak havoc through the London Underground. A message appears on the laptop screen, taunting Q, reading "Not such a clever boy".

It seems to us like Q was not aware of Joanna Rutkowska[63]'s work before getting the job of quartermaster, let alone plugging that cable. Maybe it is time for MI6 to review their hiring processes.

Unbeknownst to most filmgoers is the fact that Ms Rutkowska is at the origin of the name "Evil maid attack"[64], physically perpetrated against or through computing devices, allowing attackers to steal information or potentially even injecting malware, to be remotely activated later on.

(Kids: never plug your devices to those wall-mounted USB sockets you find on hotels and such. You have been warned. And no, we are not kidding.)

The noughties[65] were a troublesome, yet revealing, decade for computer security, dictated by a global political turmoil that would trigger an even more complicated decade right after (and let us not talk about the current one, shall we).

In terms of computer security, after learning from Microsoft itself in 2002 that most software was vulnerable[66] to exploits of a gazillion kinds, that network security[67] was almost non-existent, and that the infrastructure of our modern world is at risk of collapsing[68] every minute, the wake-up call has been violent, and the response, at best, sloppy and dull. Yes, we are now in 2025, and we have Rust, but until the great rewriting[69] is over, we will have to deal with a world run by quite deficient software, at best. While we wait for our app to compile, the state of the world keeps degrading.

It is not like we did not know any better. Just like Cassandra[70], the daughter of Priam, himself king of Troy, who was cursed with the gift of prophecy and the inability of convincing anyone of her predictions, we have had many experts warning the industry

[63] https://en.wikipedia.org/wiki/Joanna_Rutkowska
[64] https://en.wikipedia.org/wiki/Evil_maid_attack
[65] https://en.wikipedia.org/wiki/2000s
[66] https://deprogrammaticaipsum.com/microsofts-writings-on-security/
[67] https://deprogrammaticaipsum.com/sniffing-packets/
[68] https://deprogrammaticaipsum.com/david-rice/
[69] https://deprogrammaticaipsum.com/the-great-rewriting-in-rust/
[70] https://en.wikipedia.org/wiki/Cassandra_(metaphor)

of impeding disasters, to no avail. Suffice to mention Window Snyder and Kate Moussouris[71], Bruce Schneier[72], and even our dear friend Anastasiia Vixentael[73].

It is mandatory for us to add Ms Rutkowska to this illustrious list of pioneers and experts. During the infamous decade of the 2000s she published quite an impressive series of papers[74] about various research subjects, ranging from general security[75] topics, to low-level Windows kernel exploits[76], to memory forensics[77], and to USB security[78]. She is also the founder and main developer behind Qubes OS[79], described as a "reasonably secure operating system", and praised by none less than Edward Snowden[80] himself.

Needless to say, Ms Rutkowska has reached the status of a living legend in the field. And today we are going to focus our attention in one of those papers: "Intel x86 considered harmful"[81], published in October 2015.

(Spoiler alert: no, this is not about the Pentium FDIV bug[82] of 1994, already quite the damning record for Intel, if you ask me. This is much worse; sensible readers beware.)

It is, surprisingly, quite a readable paper for a software person like me, without an extended knowledge of hardware minutiæ. Her text starts with a very thoughtful and philosophical discussion about what the words "trust" and "trustworthy" mean; a fascinating topic we dedicated a whole issue[83] to last April. Why and how do we consider a particular system trustworthy? Ms Rutkowska's point of view is overwhelming and obliterating, in a quote we have shared previously[84] in this magazine:

> The word "trusted" is a sneaky and confusing term: many people get a

---

[71] https://deprogrammaticaipsum.com/on-modern-security-culture/

[72] https://en.wikipedia.org/wiki/Bruce_Schneier

[73] https://deprogrammaticaipsum.com/secure-development-is-dead-long-live-secure-development/

[74] https://blog.invisiblethings.org/papers/

[75] https://theinvisiblethings.blogspot.com/2008/09/three-approaches-to-computer-security.html

[76] https://blackhat.com/presentations/bh-usa-07/Rutkowska/Presentation/bh-usa-07-rutkowska.pdf

[77] https://docs.huihoo.com/blackhat/dc-2007/bh-dc-07-rutkowska-beyond-the-cpu-defeating-hardware-based-ram-acquisition-tools.pdf

[78] https://blog.invisiblethings.org/2011/05/31/usb-security-challenges.html

[79] https://www.qubes-os.org/

[80] https://en.wikipedia.org/wiki/Edward_Snowden

[81] https://blog.invisiblethings.org/papers/2015/x86_harmful.pdf

[82] https://en.wikipedia.org/wiki/Pentium_FDIV_bug

[83] https://deprogrammaticaipsum.com/issue/issue-079-trust/

[84] https://deprogrammaticaipsum.com/who-do-you-trust/

> warm fuzzy feeling when they read it, and it is treated as a good thing. In fact the opposite is true. Anything that is "trusted" is a potentially lethal enemy of any secure system. (...)
>
> The Operating System's kernel, drivers, networking- and storage-subsystems are typically considered trusted in most contemporary mainstream operating systems such as Windows, Mac OSX and Linux; with Qubes OS being a notable exception. This means the architects of these systems assumed none of the these components could ever get compromised or else the security of the whole OS would be devastated. (...) Quite an assumption indeed!

Well, there you go Q, you clearly trusted your MI6 systems too much. (Well, to his defense, we can say the movie was filmed in 2012 and this paper was released 3 years later.)

She argues in her paper that modern Intel x86 platforms cannot be considered trustworthy due to their complex firmware, a series of never ending and persistent vulnerabilities in the boot process, and even worse, the introduction of opaque hardware components like the Intel Management Engine[85].

(Those readers with good memory will also remember the curious fact that researchers discovered[86] in 2017 that the Intel Management Engine had an embedded version of Minix 3[87] running inside of it, a fact never fully disclosed by Intel themselves. Minix, for those unaware, is a famous "Unix-like" operating system originally written in the 1980s by professor Andrew Tanenbaum[88] to teach his operating systems class in the Vrije Universiteit Amsterdam[89]. But as usual, I digress.)

The paper then dives into security aspects of the boot process of Intel CPUs, including BIOS, UEFI, SMM, and how insecure the whole ensemble is at the end, even providing some recommendations about how to secure your Intel systems to the maximum possible extent (which, she argues, it is not much). She gave a conference talk precisely around the subject of "reasonably trustworthy x86 laptops"[90] in December 2015, right after the publication of this paper, also worth a watch.

---

[85] https://en.wikipedia.org/wiki/Intel_Management_Engine
[86] https://troopers.de/downloads/troopers17/TR17_ME11_Static.pdf
[87] https://en.wikipedia.org/wiki/Minix_3
[88] https://en.wikipedia.org/wiki/Andrew_S._Tanenbaum
[89] https://en.wikipedia.org/wiki/Vrije_Universiteit_Amsterdam
[90] https://www.youtube.com/watch?v=rcwngbUrZNg

Next comes a discussion about peripherals and their vulnerabilities, including network devices (again, Q, read this paper, please). And right after, a chapter dedicated to the Intel Management Engine, a topic she had been studying[91] and lecturing[92] about for quite a while at the time of the publication of this paper.

> Intel ME is very much similar to the previously discussed SMM. Like SMM it is running all the time when the platform is running (but unlike SMM can also run when the platform is shut down!). Like SMM it is more privileged than any system software running on the platform, and like SMM it can access (read or write) any of the host memory, unconstrained by anything.

Yes, the mere fact of plugging your laptop to the main electricity socket is enough for the Intel Management Engine to kick off. And no, it cannot be disabled, leading to what Ms Rutkowska calls the "zombification" of general-purpose operating systems, and to the *de facto* existence of what she calls "an ideal rootkiting infrastructure":

> When reading through the "ME Book" it is quite obvious that Intel believes that 1) ME, which includes its own custom OS and some critical applications, can be made substantially more secure than any other general purpose system software written by others, and 2) ultimately all security-sensitive computing tasks should be moved away from the general purpose OSes, such as Windows, to the ME, the only-one-believed-to-be-secure-island-of-trust...
>
> There is another problem associated with Intel ME: namely it is just a perfect infrastructure for implanting targeted, extremely hard (or even impossible) to detect rootkits (targeting "the usual suspects").

You get the idea. I can only recommend diving into this fascinating text (particularly if you land the job of quartermaster at MI6) and, well, maybe choosing another architecture than x86 for your next laptop, if all else fails. Because according to Ms Rutkowska, the situation is not really different with AMD processors:

> And it seems AMD has an equivalent of Intel ME also, just disguised as Platform Security Processor (PSP).

---

[91] https://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf

[92] https://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20slides.pdf

To finish this joyful edition of *De Programmatica Ipsum* dedicated to the topic of "Considered Harmful", we can recommend Dijkstra's own "Go To Statement Considered Harmful"[93]; then, continue with " 'Stored Program Concept' Considered Harmful: History and Historiography"[94] by Hutchinson et al. Let us not forget about "Debunking the 'Expensive Procedure Call' Myth, Or Procedure Call Implementations Considered Harmful, Or Lambda: The Ultimate GOTO"[95] by Guy Steele Jr. Also, "Recursive Make Considered Harmful"[96] by Peter Miller, "Prototyping Considered Dangerous"[97] by Michael Atwood et al., "Global Variable Considered Harmful"[98] by Wulf and Shaw, "GOFAI Considered Harmful"[99] by Drew McDermott, and "Electron considered harmful"[100] by Drew DeVault.

After reading these pieces, however, you will have the eerie feeling that pretty much anything related to computers should be considered harmful. And you might be right. *Le sigh*.

Cover photo by the author.

[93] https://dl.acm.org/doi/10.1145/362929.362947
[94] http://link.springer.com/10.1007/978-3-642-39053-1_28
[95] https://ia802904.us.archive.org/17/items/bitsavers_mitaiaimAI_1976403/AIM-443_text.pdf
[96] https://aegis.sourceforge.net/auug97.pdf
[97] https://link.springer.com/chapter/10.1007/978-1-5041-2896-4_30
[98] https://dl.acm.org/doi/10.1145/953353.953355
[99] https://www.cs.yale.edu/homes/dvm/papers/nogofai.pdf
[100] https://drewdevault.com/2016/11/24/Electron-considered-harmful.html