Issue 037: Microsoft

Adrian Kosmaczewski

October 4th, 2021

Welcome to the thirty-seventh issue of *De Programmatica Ipsum*, opening our fourth year of publication, and dedicated to *Microsoft*. In this edition:

- Graham explains how programmers have perceived[1] Microsoft and its success through the years.
- Adrian reviews 45 years of Microsoft history[2] and finds patterns, anecdotes, and some lessons.
- In the Library section[3], Graham analyzes Microsoft's books about computer security[4].

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

Cover photo by Clint Patterson[7] on Unsplash[8].

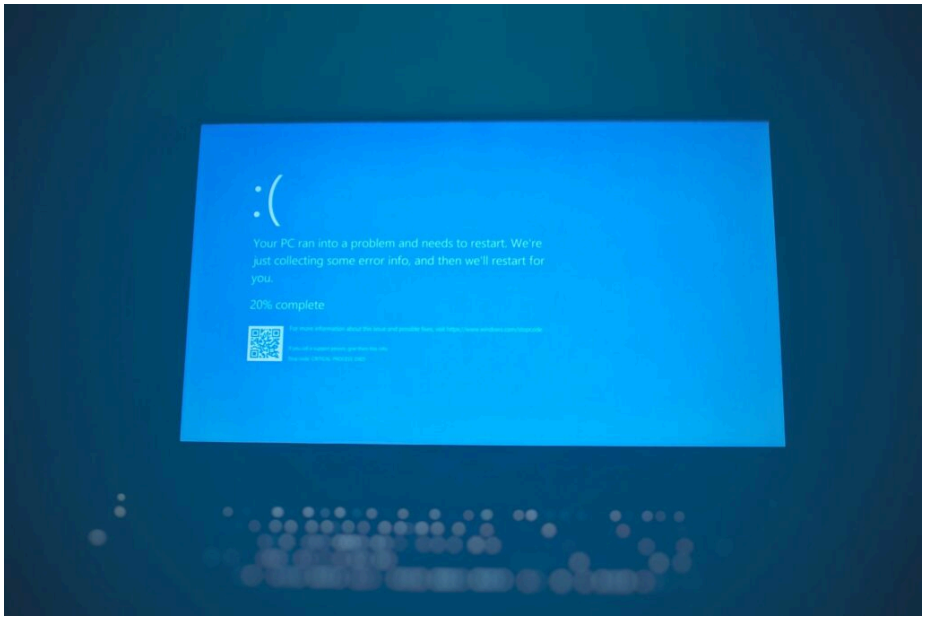
---

[1] https://deprogrammaticaipsum.com/putting-the-dollar-sign-in-microsoft/
[2] https://deprogrammaticaipsum.com/where-does-microsoft-want-to-go-today/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/microsofts-writings-on-security/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@cbpsc1?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/microsoft?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Putting The $ In Micro$oft

Graham Lee

October 4th, 2021



When I started in the world of professional computing, it was popular to spell MS as micro*oftormicro*haft or micro$loth. Or, if you were feeling particularly snarky, micros~1. Windows was invariably spelt windoze. What did they do to deserve that reputation?

Microsoft started life as a developer tools company: they wrote the first BASIC on the MITS Altair, and licensed BASICs on most micros of the 1970s and 1980s (including the Apple II, even though Steve Wozniak had written Applesoft BASIC for it already). That is where I learned to program, in Microsoft Extended Color BASIC on the Dragon 32[1] (a British rip-off of the Tandy Color Computer).

But that means that they are one of the many US computer companies that got rich by stealing public sector inventions, enclosing them, and profiting from them. Just as Cisco and SUN took the work that had been done at Stanford University, Microsoft took the BASIC language invented at Dartmouth without a license. They actually developed that Altair BASIC on a PDP-10 operated by the military and hosted at Dartmouth, and MS did not pay for their computer time. I hope all those American taxpayers are proud that their money was funnelled directly to someone who would become the world's richest man, just as it is now going to Jeff Bezos via the NASA budget!

Microsoft *did*, I suppose, innovate[2], in that they added PEEK and POKE commands to BASIC. Indeed perhaps Bill Gates himself first implemented those. In these days where the

[1] https://worldofdragon.org/index.php?title=Dragon_32
[2] https://deprogrammaticaipsum.com/issue-36-innovation/

UK government wants all ballerinas to retrain in cyber[3] we might be appalled at such an obvious way to bypass, well, everything, and expect that this was what brought the ire of the developer community.

But no. There was not really anything to bypass on a microcomputer with an 8-bit Intel 8080 or equivalent processor and a few kilobytes of RAM. The only thing you were "working around" was the fact that not all of the hardware had a BASIC command: maybe you could read keypresses with INKEY$ but to read the joystick direction you had to directly look at a byte in memory.

Microsoft may not have believed much in paying the people who invented their products (see not only BASIC, but also MS-DOS), or supplied their resources, but they definitely believed in everybody else paying them. Bill Gates's Open Letter to Hobbyists[4] circulated to members of the Homebrew Computer Club and more widely via various magazines first introduces the idea of "sharing" software, then rewrites this to be theft.

> As the majority of hobbyists must be aware, most of you steal your software.

Gates and Allen had organised a royalty model for licensing Altair BASIC to MITS. If they had charged a fixed development fee instead then there would not be any concept of software piracy, no over-the-top Genuine Windows™ holograms, and no unactivated copies of Windows. Software companies would be paid for writing software, like plumbers are paid for doing plumbing: they would not be paid for having written software, like record labels are paid for selling *Dark Side of the Moon* over and over on different formats.

This interest in extracting every possible cent from customers, and creating customers where none exist, could certainly be a source of the cynicism displayed when nerds write the $ in micro$oft. You can see it today, in Windows 10 and its "Windows as a Service" mentality. Having sold your OEM a license for you to use Windows, and potentially sold you an update license too, it is then time for the upsell. There are ads in the Solitaire app, and an in-app purchase to remove them. From Windows 3.0 to Windows 7, it was a free distraction; now it is a revenue stream. The archive manager is shareware and you can pay an annual subscription to unlock additional features like supporting some archive formats.

Microsoft's tempestuous relationship with open source perhaps plays no little part in stoking the anti-Microsoft sentiment among nerds. The famous collection of Halloween Documents[5] curated by Eric S. Raymond show a company first trying to understand, then come to terms with, then compete with, then undermine, not so much another product but another way of thinking about ownership completely at odds with the Microsoft way and "most of you steal your software". No we do not, we are free to use the software for any purpose!

In the time since I was one of the interchangeable goons of the undiverse free software hoards, in my long hair, black band t-shirt, and combat trousers, Microsoft's relationship with free software has changed significantly. They always used the software for any purpose where that purpose suited: various of the networking utilities in Windows were ported from BSD (which of course, the free software BSD license permits; if free software developers are angry about this, they should perhaps choose a different license such as the Affero GPL).

The fully-proprietary Windows Services for UNIX has been replaced, first by the Linux-compatible Windows Services for Linux, then by WSL2 which is actually real GPL2-licensed

[3] https://www.standard.co.uk/news/uk/fatima-ballet-dancer-job-cyber-government-campaign-a4568641.html

[4] https://en.wikipedia.org/wiki/Open_Letter_to_Hobbyists

[5] http://www.catb.org/~esr/halloween/

Linux running in a virtual machine. Microsoft has bought GitHub, the software forge website that loves open source so much it is a fully proprietary product. The one that hoovers up all that source code[6] to feed the mighty Copilot, so that you can accidentally paste AGPL3 code into your proprietary application[7] and hide behind "the AI did it".

They sink significant resources into various open source projects: TypeScript, npm, Xamarin, Visual Studio Code, even the Common Language Runtime and .NET. Further than that, their browser, once the scourge of front-end engineers the world over, is now based on the open source Chromium project.

This all looks like it could be the start of another round of "embrace, extend, and extinguish", a business model popular among Microsoft management. Readers who were not around the software sector in the 1990s-2000s may not be aware that Microsoft's approach to competition used to be to integrate with the competing products, preferably using open standards, then break the integration until everybody needed things that worked the broken way (i.e. they needed Microsoft's products). A combination of making Internet Explorer free, and making Internet Explorer incompatible with Netscape Navigator, eventually killed Netscape. Outlook and Exchange work with all of the usual email and groupware standards, except in the ways that they do not.

Anyway how does this embrace, extend, extinguish idea apply to open source? Microsoft have worked out that by providing the tools people need for the development they are really doing, rather than the old model of the development Microsoft wished they were doing, they can get incredibly detailed and accurate information about a large chunk of the developer community: think of the analytics from GitHub combined with the telemetry from Visual Studio Code and its array of plugins.

They can use this to get an advantage over competitors by getting information about how the competitors' products are used, in addition to their own. And in addition to what open source libraries people are using to augment those products; i.e. what features might be missing. Somewhere on Satya Nadella's F: drive there will be a spreadsheet showing up to the minute data on whether Microsoft should be adding a souped-up version of leftpad or isnumber to the premium tier of MSDN.

So surely all of this, the way that Microsoft is adding their proprietary telemetry to the galaxy of open source software, is enough to warrant the opprobrium? Have they finally earned the $ in Micro$oft?

Let us be honest. The real reason that dollar sign appeared and the nicknames abounded was that Microsoft were good at what they did, and their products were popular. Yes, they absolutely did use underhand business tactics to get as popular as they did: crowding out competing browsers and media players (tell me how easy Apple have made it for competitors to integrate browsers and media players on the iPhone, I will wait); pressuring OEMs to abandon alternative operating system vendors; breaking interoperability and compatibility with competing products.

They also did a good—maybe not perfect, but certainly very good—job of walking the tightrope over the maelstrom that is the conflict between their own short term interests, and the long term interests of their platform: keeping the users and the developers as mutually happy as possible.

[6] https://zephyrtronium.github.io/articles/copilot.html
[7] https://fossa.com/blog/analyzing-legal-implications-github-copilot/

Yes, if you built an office suite that got an appreciable market share, you would find yourself on the receiving end of some very dubious competition, though not as outright evil as has been claimed; the idea that Microsoft lived by the mantra "DOS ain't done 'til Lotus won't run" has been soundly debunked[8]. This belongs in the same dustbin of fake MS news as the 640 KB RAM is enough for anybody[9] quote.

But for everybody else, you might find yourself on stage with Microsoft's developer division, as one of their most valued professionals. Sure, the MVP program almost disappeared[10], but only for a weekend.

Microsoft understands that developers are not their enemy, for the most part. That they need third party software to add value to the Windows galaxy. Their job is not to ban developers from the platform for making money that did not go to the first-party apps; but to help everybody else help make Windows more useful. So if you go to a Microsoft developer conference, you will find authors of books for Microsoft Press[11] who do not work for Microsoft. You will find interviews with third-party developers being recorded for Channel 9[12].

All that success breeds resentment. What do you do if you are in the Free Software galaxy, and far from eating the world you are mopping up the scraps that the incumbents drop from their table? If you are good, you explain how what you offer is better. If you are patient, you explain how the way you do things is better. And if you are vindictive, you fling insults.

People are angry at Microsoft not so much for the *way* they succeeded, but because they are successful *at all*. That, then, is the source of the $ in micro$oft.

Photo by Joshua Hoehne[13] on Unsplash[14].

[8] https://web.archive.org/web/20100104083410/http://www.proudlyserving.com/archives/2005/08/dos_aint_done_t.html

[9] https://www.computerworld.com/article/2534312/the--640k--quote-won-t-go-away----but-did-gates-really-say-it-.html

[10] http://www.mvps.org/about/kissoff.html

[11] https://www.microsoftpressstore.com

[12] https://channel9.msdn.com

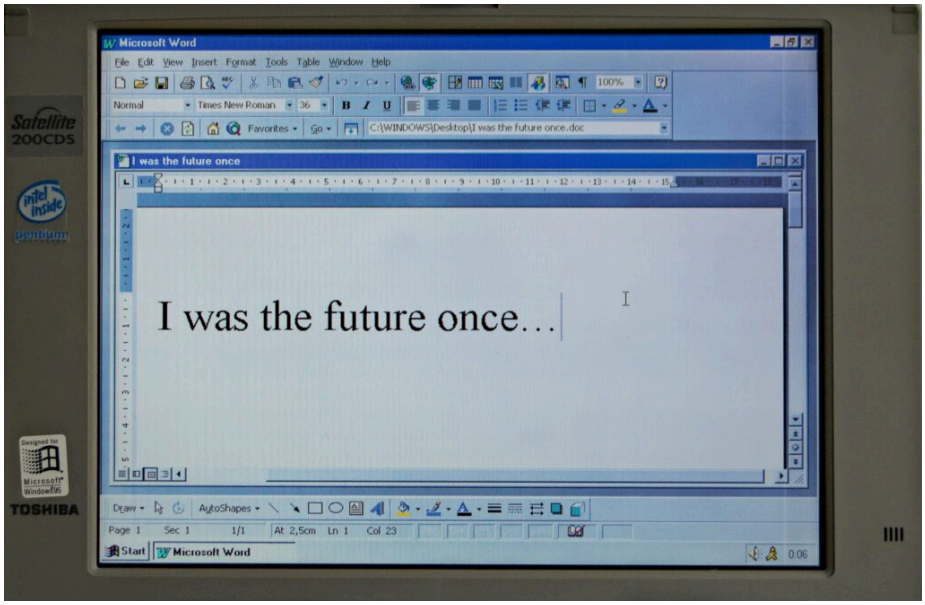
[13] https://unsplash.com/@mrthetrain?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[14] https://unsplash.com/s/photos/blue-screen?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Where Does Microsoft Want To Go Today?

Adrian Kosmaczewski

October 4th, 2021



In May 30th, 2007, Kara Swisher and Walt Mossberg interviewed[1] Bill Gates and Steve Jobs in a now widely cited D5 conference panel. At the beginning of that session, both were asked about what they thought was the greatest contribution the other made to the computer industry. Speaking about Bill Gates, Steve Jobs said:

> Well, you know, Bill built the first software company in the industry and I think he built the first software company before anybody really in our industry knew what a software company was, except for these guys.

A software company. Does that short phrase still describe the Microsoft of today?

A quick search[2] of the word "Microsoft" in this publication brings around 40 articles: yes, around one third of all articles in the website you are reading right now mention the company in one way or another. But they have not always been this relevant in the industry.

Actually, it is fair to say they spent at least the first five years of their history in a relative obscurity. The first appearance of the name "Microsoft" on the pages of Byte Magazine was a short mention to their BASIC interpreter in an advertising of Ohio Scientific Instruments in page 47 of the May 1977 issue[3]. In sharp contrast, page 34 of the very same issue[4] includes a complete description of the Apple II by Steve Wozniak himself. The Woz and Gary Kildall[5] were undoubtedly the biggest stars of the burgeoning hobbyist microcomputer industry at the end of the 1970s.

[1] https://allthingsd.com/20070531/d5-gates-jobs-transcript/
[2] https://deprogrammaticaipsum.com/?s=microsoft
[3] https://archive.org/details/byte-magazine-1977-05/page/n47/mode/2up
[4] https://archive.org/details/byte-magazine-1977-05/page/n35/mode/2up
[5] https://en.wikipedia.org/wiki/Gary_Kildall

To add insult to injury, the December 1975 issue of the same Byte Magazine, dedicated to the MITS Altair 8800, contains an opinion piece[6] mentioning a "proprietary software product (BASIC) aimed directly at the microcomputer hobbyists". Of course, that was precisely Microsoft's first product[7]; but Chris Ryland, the author of the article, did not even mention them, as MITS was distributing it under a license.

Interestingly enough, Mr. Ryland's article title was "The Software Vacuum". In hindsight, one could easily think that Microsoft set themselves to fill that software vacuum, come what may.

It was not until Microsoft struck a deal with IBM[8] in November 1980, leading to the release of the first IBM PC, that their name started becoming ubiquitous in this industry. Since that point on, the company has evolved in three major cycles, each corresponding to the person at its helm.

The first cycle, led by Microsoft's first CEO and founder, Bill Gates, was characterized by some famed memos and initiatives: "Information at your Fingertips"[9] in 1990 (in a video where he looks like Max Headroom[10]); "The Internet Tidal Wave"[11] in 1995 (where he corrected the mistake of barely mentioning the Internet in his book "The Road Ahead"[12]); and "Trustworthy Computing"[13] in 2002 (described in detail by Graham in this month's Library article[14]). Those were a times where a tasteless[15] Microsoft asked the Rolling Stones to add some music to a wildly successful product launch[16].

Bill Gates also made famous appearances on stage, for example demoing important Windows 98 features[17], or being booed by Apple enthusiasts[18] even though he literally saved Apple from a near death experience–simultaneously bringing some positive karma to his own sorry company, during the heated Congress antitrust hearings[19] being held at the time, that is.

The second cycle, led by Steve Ballmer[20], is mostly remembered by blunders[21] and his unmoderated love for the word "developer"[22]. Joel Spolsky quickly realized in 2005 how far Microsoft had fallen behind Google[23]:

> The very fact that Google invented MapReduce, and Microsoft didn't, says something about why Microsoft is still playing catch up trying to get basic search features to work, while Google has moved on to the next problem: building Skynet$^{H}H^{H}H^{H}H$ the world's largest massively parallel supercomputer. I don't think Microsoft completely understands just how far behind they are on that wave.

[6] https://archive.org/details/byte-magazine-1975-12/page/n13/mode/2up
[7] https://en.wikipedia.org/wiki/Altair_BASIC
[8] https://thisdayintechhistory.com/11/06/ibm-signs-a-deal-with-the-devil/
[9] https://www.youtube.com/watch?v=tWd8DxLfDek
[10] https://en.wikipedia.org/wiki/Max_Headroom_(TV_series)
[11] https://www.wired.com/2010/05/0526bill-gates-internet-memo/
[12] https://en.wikipedia.org/wiki/The_Road_Ahead_(Gates_book)
[13] https://news.microsoft.com/2012/01/11/memo-from-bill-gates/
[14] https://deprogrammaticaipsum.com/microsofts-writings-on-security/
[15] https://www.youtube.com/watch?v=rsyOlwmHt5E&t=4s
[16] https://www.youtube.com/watch?v=y0CRWAz09r8
[17] https://www.youtube.com/watch?v=IW7Rqwwth84
[18] https://youtu.be/WxOp5mBY9IY?t=285
[19] https://archive.nytimes.com/www.nytimes.com/library/cyber/week/110597senate.html
[20] https://www.youtube.com/watch?v=hnUnxHaq6H8
[21] http://www.asymco.com/2013/08/26/steve-ballmer-and-the-innovators-curse/
[22] https://www.youtube.com/watch?v=VM-2OVNt-eQ
[23] https://www.joelonsoftware.com/2005/12/29/the-perils-of-javaschools-2/

However dire the situation was, the same Spolsky was the first to acknowledge in 2004[24] that

> Microsoft has an incredible amount of cash money in the bank and is still incredibly profitable. It has a long way to fall. It could do everything wrong for a decade before it started to be in remote danger, and you never know... they could reinvent themselves as a shaved-ice company at the last minute.

And the company did almost everything wrong for literally a whole decade. Laughing about the iPhone[25] and then regretting that laugh seven years later to Charlie Rose[26] did not help Windows Phone to survive. At least they did not miss on the cloud train, something that IBM did[27]. It was the time of the Mini-Microsoft blog[28], written by a still anonymous insider[29]. Those were the years of "Linux is Cancer"[30] with Microsoft throwing fuel[31] at the SCO-Linux dispute with an infamous "Get the Facts" advertising campaign[32]. That was the age of Internet Explorer 6 bringing the web to a standstill[33] for a decade.

The third cycle is the one we are witnessing as this article hits the press, with Satya Nadella in charge. The age of Microsoft buying GitHub[34] and npm[35]. The moment of TypeScript[36] eating JavaScript for breakfast. The time of SQL Server on Linux[37]. The years of Visual Studio Code[38] becoming a de-facto standard.

It is also the moment of Microsoft being a member of literally every single software foundation you can think of. Let us now enumerate them, shall we, so that we all understand the level of change. In 2016 they shocked[39] pretty much everyone by joining the Linux Foundation[40] and its many related bodies: the CNCF[41], OpenChain[42], Hyperledger[43], OpenJS[44], and just a few months ago, the eBPF Foundation[45].

Nowadays Microsoft sponsors both the Apache Foundation[46] and the Python Software Foundation[47]. It is also a full member of the Unicode Consortium[48], the ISO C++

[24] https://www.joelonsoftware.com/2004/06/13/how-microsoft-lost-the-api-war/
[25] https://www.youtube.com/watch?v=eywi0h_Y5_U
[26] https://www.youtube.com/watch?v=v9d3wp2sGPI
[27] https://www.protocol.com/enterprise/ibm-lost-public-cloud
[28] https://minimsft.blogspot.com/
[29] https://twitter.com/whodapunk
[30] https://www.theregister.com/2001/06/02/ballmer_linux_is_a_cancer/
[31] https://en.wikipedia.org/wiki/SCO%E2%80%93Linux_disputes#Microsoft_funding_of_SCO_controversy
[32] https://www.zdnet.com/article/microsoft-kills-its-get-the-facts-anti-linux-site/
[33] https://techcrunch.com/2021/05/20/so-long-internet-explorer-and-your-decades-of-security-bugs/
[34] https://news.microsoft.com/announcement/microsoft-acquires-github/
[35] https://itsfoss.com/microsoft-npm-acquisition/
[36] https://www.typescriptlang.org/
[37] https://blogs.microsoft.com/blog/2016/03/07/announcing-sql-server-on-linux/
[38] https://code.visualstudio.com/
[39] https://arstechnica.com/information-technology/2016/11/microsoft-yes-microsoft-joins-the-linux-foundation/
[40] https://www.linuxfoundation.org/join/members/
[41] https://www.cncf.io/about/members/
[42] https://www.openchainproject.org/about
[43] https://www.hyperledger.org/about/members
[44] https://openjsf.org/about/members/
[45] https://ebpf.io/members/
[46] https://apache.org/foundation/thanks
[47] https://www.python.org/psf/sponsorship/sponsors/
[48] https://home.unicode.org/membership/members/

Committee[49], the Rust Foundation[50], the MariaDB Foundation[51], the Eclipse Foundation[52], the Open Source Security Foundation[53], the W3C[54], the Bytecode Alliance[55] for WebAssembly, the Blender Development Fund[56], the R Consortium[57], the Academy Software Foundation[58]... and the F# Foundation[59]. Well this last one was kind of obvious, I give you that.

As the adage says, if you cannot beat them, join them. Or maybe is this just a new chapter of their traditional embrace, extend, and extinguish[60] strategy? Only time will tell.

In the aforementioned 2007 panel that opens this article, Steve Jobs also said about Bill Gates:

> I think the world's a better place because Bill realized that his goal isn't to be the richest guy in the cemetery, right?

Thanks to Bill Gates, and his newly found retirement hobby of saving the world, Microsoft owns 24 million shares[61] of the Bill & Melinda Gates Foundation[62], and is even a member of the Climate Finance Foundation[63], whatever that is.

The one tradition that remained constant throughout all of Microsoft's history is their infatuation for the BASIC programming language, through its various declinations: the already mentioned Altair BASIC[64], Microsoft BASIC[65], GW-BASIC[66], MBASIC[67], MSX BASIC[68], Commodore BASIC[69], QuickBASIC[70], QBasic[71], the "classic" Visual Basic[72], VBA[73], VBScript[74], VB.NET[75], and finally Small Basic[76]; the latter available either on your Surface laptop or, even more 21st century-like, on Azure[77] and best viewed through the Microsoft Edge browser.

---

[49] https://isocpp.org/std/the-committee
[50] https://foundation.rust-lang.org/
[51] https://mariadb.org/about/
[52] https://www.eclipse.org/membership/exploreMembership.php
[53] https://openssf.org/about/members/
[54] https://www.w3.org/Consortium/Member/List#xM
[55] https://bytecodealliance.org/
[56] https://fund.blender.org/
[57] https://www.r-consortium.org/members
[58] https://www.aswf.io/members/
[59] https://foundation.fsharp.org/
[60] https://en.wikipedia.org/wiki/Embrace%2C_extend%2C_and_extinguish
[61] https://en.wikipedia.org/wiki/Bill_%26_Melinda_Gates_Foundation#Trust_investments
[62] https://www.gatesfoundation.org/
[63] https://www.linuxfoundation.org/press-release/2020/09/new-lf-climate-finance-foundation-to-host-open-source-initiative-to-address-climate-risk-and-opportunity-in-financial-sector/
[64] https://en.wikipedia.org/wiki/Altair_BASIC
[65] https://en.wikipedia.org/wiki/Microsoft_BASIC
[66] https://en.wikipedia.org/wiki/GW-BASIC
[67] https://en.wikipedia.org/wiki/MBASIC
[68] https://en.wikipedia.org/wiki/MSX_BASIC
[69] https://en.wikipedia.org/wiki/Commodore_BASIC
[70] https://en.wikipedia.org/wiki/QuickBASIC
[71] https://en.wikipedia.org/wiki/QBasic
[72] https://en.wikipedia.org/wiki/Visual_Basic_(classic)
[73] https://en.wikipedia.org/wiki/Visual_Basic_for_Applications
[74] https://en.wikipedia.org/wiki/VBScript
[75] https://en.wikipedia.org/wiki/Visual_Basic_.NET
[76] https://en.wikipedia.org/wiki/Microsoft_Small_Basic
[77] https://superbasic-v2.azurewebsites.net/

Cover photo by Pedro Santos[78] on Unsplash[79].

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

[79]https://unsplash.com/s/photos/microsoft?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Microsoft's Writings On Security

Graham Lee

October 4th, 2021

Yes, you read that correctly. Microsoft. Writing on information security. They may be the software company who have done the most writing on information security, including many security software companies.

The reason they did it is one of those things that made Bill Gates such a capable and successful business leader. He was not necessarily a product visionary, though he did come up with the "a computer on every desktop" vision that made Windows and Office ubiquitous. No, Windows and Office were not the first examples of products in their field to market. What Bill Gates was great at was seeing what he and his team were not great at, and doing that.

In 2002, this was security. Various worms and viruses plagued the Windows ecosystem. These had names that we recognise now, like Nimda and Code Red, which helped to cement their place in the public consciousness in those as Spectre and Meltdown have elevated the publicity of some CPU design flaws today. The "ILOVEYOU" virus even got debated in the House of Commons[1] of the UK parliament, as a national defence issue.

Microsoft could probably have done nothing, and been OK for a while. Through a combination of an improved product and shady practices—pricing OEM licenses unfavourably if the vendors offered alternative operating systems—Windows 95 had already won an overwhelming share of the desktop computer market, a position which Windows XP had solidified.

[1] https://hansard.parliament.uk/Commons/2000-07-04/debates/2aee8540-6e7f-4fdb-b620-0b8b6e410298/ComputerViruses?highlight=love#contribution-9d212745-cadb-44b3-a256-55ce707b7c09

When you run the whole market it can be difficult to see a need to change direction, but Bill Gates could do it. He did it in 1995 with his Internet Tidal Wave[2] memo: Microsoft were late to the party, but they were going to show up in a massive stretch Hummer and spike the punch.

He did it again in 2002 with his Trustworthy Computing[3] memo.

> Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. If we do not do this, people simply will not be willing – or able – to take advantage of all the other great work we do. Trustworthy Computing is the highest priority for all the work we are doing. We must lead the industry to a whole new level of Trustworthiness in computing.

Software engineers across the whole of Microsoft downed tools (there was no new development on Windows for two months), learned how to build things in a security-conscious fashion, then invested time into building threat models, identifying risks, and patching their software. Never before or since has a software company so totally—and publicly—admitted its faults, and come to a halt while it addresses them. And along the way, we got some useful information about how to do the same ourselves.

> We've also published books like "Writing Secure Code," by Michael Howard and David LeBlanc, which gives all developers the tools they need to build secure software from the ground up.

So BillG even has some recommended reading for us. Shall we take a look?

Recommended for us but required reading for Microsoft engineers, *Writing Secure Code* explains why secure systems are needed, gives a light touch description of threat modeling, then goes into a load (440 pages) of secure coding techniques. Many of these techniques are still needed today, because they are design-level (access control, execution privilege, appropriate application of cryptography) or because we never learn (buffer overruns, data representation, code injection). For example, their description of "Canonical Representation Issues" covers the various ways in which Unicode text can look like other Unicode text, a problem in 2002 that foreshadows the rise of punycode domain spoofing that still plagues the Firefox browser in 2021.

Why are these things, these important security aspects of application and system design, still a problem two decades later? Why is Anastasiia Vixentael, the guest author in our security issue[4], still explaining to developers how to do (and that they should do) these things that were known back before many React Native coders were even conceived?

The answer is found in the excellent and evergreen Appendix B to Howard and LeBlanc's book, "Ridiculous Excuses We've Heard". This covers reasons Microsoft teams gave the authors why they should not need to adopt secure coding practices, along with counter-arguments to each of them. Yes, I have been the information security expert on many a team in the 21st century, and yes, I have heard many of these ridiculous excuses.

Of course the ridiculous excuses are there to cover the vulnerability of the developers, project managers, product owners, testers, and other engineers: they were never taught about information security or secure development practices. Telling them it is the most important thing

[2] https://www.wired.com/2010/05/0526bill-gates-internet-memo/
[3] https://news.microsoft.com/2012/01/11/memo-from-bill-gates/
[4] https://deprogrammaticaipsum.com/secure-development-is-dead-long-live-secure-development/

to the company is telling them their most pressing need is one they are ill-equipped to handle, and they get defensive. In the introduction to Writing Secure Code, Howard and LeBlanc make the simple observation that the most popular MS Press book on programming, Code Complete 2[5], does not mention security once.

I will save you some trouble by modernising that statement: programmers still are not told a lot about security as part of software development. *Succeeding with Agile* does not tell you to adopt secure coding practices, or build security into your backlog. *Continuous Delivery* tells you that "capacity and security requirements" might exist multiple times, but its only specific recommendation is to use a linter to catch common problems. *The Pragmatic Programmer* does have a topic, "stay safe out there", explaining basic security principles. The Pragmatic approach, apparently, is to let somebody else worry about it. Let us hope that some engineers never read that book, or we are a generation or two away from everybody hoping that somebody else is "doing security" for them.

Michael Howard also co-authored a book with Steve Lipner, also in the "Microsoft secure software DEVELOPMENT SERIES" as the cover styles it, called *SDL: The Security Development Lifecycle*. This book could be treated as a timeless guide to designing security into a software development process. There is a list of banned functions that you might want to update, and a Visual C++ solution on a companion CD that will only work in Visual Studio 2005 apparently, but other than that everything in the book is about how you incorporate security thinking into the other thinking you are doing.

Thinking does not change that much. We have already seen that specific code-level problems identified in *Writing Secure Code* are still problematic today. So, it turns out, are brain-level problems. Even at ISO 27k-certified companies (the standards series for information security management systems) you will find developers who think that because they are using JWTs, they are secure[6]; and product owners who see all that data coming in and ask who it can be sold to.

The authors of the SDL take us through everything we need to do to integrate security thinking into our software efforts. Starting with before the project kicks off (education, awareness, leadership support), to evaluating whether an SDL is needed, designing security requirements, building and testing them, and making sure software is secure in deployment. They even tell you how to integrate the SDL into those new-fangled Agile™®□© methods everyone was raving about in 2002. The description is golden:

> We've heard people claim that <*insert popular development method*> produces bug-free software. This might be true—and of course, it is true if you know nothing about security bugs, because you wouldn't recognise a security bug if you had no idea what one was.

There is a bit missing from our story, though. We know how to plan the work into our project, and we know how to address code-level issues, but how do we know what the risks are? How do we understand what the security posture of our app is?

For that, we turn to the third and final of today's readings, *Threat Modeling* by Frank Swiderski and Window Snyder. Snyder[7] is a bit of a legend in software security circles. She was at Microsoft for the trustworthy computing initiative, and has held security leadership positions at Apple, Square, Fastly, Intel, and Mozilla (where her title was CSSO: "Chief Security

[5] https://deprogrammaticaipsum.com/steve-mcconnell/
[6] https://jwt.io
[7] https://twitter.com/window

Something-or-Other"). She is currently working on a startup, Thistle Technology, working to improve security of IoT[8].

The benefit of the *Threat Modeling* approach is that it is easy to understand and apply. Swiderski and Snyder have handy mnemonics for remembering important ideas. Threats are classified based on their effect with the STRIDE acronym: Spoofing, Tampering, Repudiation, Denial of Service, or Elevation of Privilege. Vulnerabilities and risks are ranked based on their DREAD scores: Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability.

Elsewhere, the threat modeling process uses skills your team probably already has. How do you identify whether an attacker can leverage some asset using some particular entry point? Build a data flow diagram; your engineers probably already do this on a whiteboard whenever they need to discuss the behaviour of some API call. How do you know what is there to be attacked? Create use scenarios: you probably already have these, and they are probably written in the form "AS A journalist, I WANT TO...".

Unfortunately Microsoft Press does not carry an up to date equivalent of any of these books. The second edition of Writing Secure Code, from 2003, is still the edition you can buy on their store. You will not find the other two at all; not even the SDL book which only needs someone to port the Visual Studio solution to a modern IDE.

We know that security problems have not gone away, so why are not Microsoft still providing leadership here? Maybe they think their online security development lifecycle guide[9] is sufficient, even though nearly half of the resources are in the Legacy Archive[10]. Maybe thinking about security was a fad, and the people involved are now all thinking about reasoning in type systems or shoveling even more unaudited javascript into everybody's computers at Microsoft subsidiary npm, Inc. Maybe the original Trustworthy Computing push achieved its goal, and moved the heat onto other vendors.

Whatever the case, software security has not gone away, and its discussion should not be limited to black hat nerds and CISSPs in suits opening calc.exe on each other's laptops in DefCon meetings. Do yourself and your customers a favour, and find copies of these excellent works in your favourite second hand book store.

Image by the author.

[8] https://techcrunch.com/2021/04/22/thistle-technology-seed-security-iot/?guccounter=1

[9] https://www.microsoft.com/en-us/securityengineering/sdl/

[10] https://www.microsoft.com/en-us/securityengineering/sdl/resources