Issue 003: Security

Adrian Kosmaczewski

December 3rd, 2018

Welcome to the third issue of *De Programmatica Ipsum*, dedicated to the subject of *Security*. In this edition:

- Our guest writer Anastasiia Vixentael[1] exposes the dramatic state of security practices[2] among software developers.
- Adrian talks about the weakest link in software security[3], asking some questions about society and ethics.
- In this issue's subscriber-only article, Graham provides a thorough review of current trends in software security[4].

Enjoy this issue! Please let us know if you have any feedback[5] and get our free newsletter[6] to stay updated about new releases. If you want to support us, subscribe[7] for a month or a year, and let us know if you would like to write with us[8].

Cover photo by shotinraww[9] on Unsplash[10].

---

[1] https://deprogrammaticaipsum.com/user/vixentael/
[2] https://deprogrammaticaipsum.com/secure-development-is-dead-long-live-secure-development
[3] https://deprogrammaticaipsum.com/the-weakest-link/
[4] https://deprogrammaticaipsum.com/on-modern-security-culture/
[5] https://deprogrammaticaipsum.com/feedback/
[6] https://deprogrammaticaipsum.com/newsletter/
[7] https://deprogrammaticaipsum.com/subscribe/
[8] https://deprogrammaticaipsum.com/write-with-us/
[9] https://unsplash.com/photos/Zw2-HhnCV2U?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[10] https://unsplash.com/search/photos/security?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Secure Development Is Dead, Long Live Secure Development

Anastasiia Vixentael

December 3rd, 2018

Does not it feel like the world is on fire?[1] Security talks and blog posts usually start with horror stories – for instance "application security is important because without it you will be hacked here and now." But I have had enough of this doom and gloom on my Twitter feed and mailbox, and would like to talk about something else.

Only those who (happily) live under a rock may have missed the latest news. One company has a critical security bug, another has leaked emails and passwords to millions of user accounts *(yours and mine included,)* and another will be fined millions of Euros due to a GDPR violation.

I do not want to add to the existential dread, but I would love to discuss the problems behind creating secure software.

Why Do We Care?

Thirty years ago, before the Internet became widespread, only military and governmental services were interested in data protection. Back then it was as complicated to gather and analyze data as it was to steal it. Today more than 1300 new apps emerge in app stores every day; most of them collecting, processing, and transfering data – risking its confidentiality and integrity at each step.

Let's put on our app developer's hat. Risking our customers' data, those who trust us to handle it with great care, is more than a question of it being "good or bad." Why? Because of

[1] https://www.schneier.com/blog/archives/2018/11/more_spectremel.html

the lawsuits, fines, and other more terrifying consequences that may loom over the business that we build, or are employed by. Competitors or attackers will be thrilled to discover you've left them an entryway, too.

Protecting users' data and privacy is not only a sign of product quality. It is also a way of showing respect for those users and their rights. One could argue that data security has a "a negative value"[2], meaning that it requires a lot of effort to implement, it's hard to measure, it's never "completely" done, and it doesn't bring money to the business. However, while "positive security" is very hard to define, the lack of regular "negative" security is something that companies cash out for in terms of fines, which may even lead to the death of a business.

No company is "too small" to think about security. "Oh, we won't be hacked because our application is small and not very popular" is a poor argument. I'm sure the owner of a Winnipeg mattress store[3] thought so too – before the company was forced to pay a criminal who shut down their servers, stopping all the sales.

A Bit Of Background

Do not you have a feeling that developers care more about smooth animations than about data protection? I think the reason might be the gap between the world of product makers and the world of security people. The gap in their skills, competence, and mindset.

I came to work in the area of security and cryptography after leaving the shiny world of mobile development. I was working in a "software boutique" company, creating iOS applications, and NodeJS or Python-driven backends. I managed to put my hands on several dozens of mobile apps, like chats, online shops, medical platforms that process patients' data, apps for controlling smart devices, and so on. We mostly worked with startups and small companies, and we didn't have a separate role of a solution architect or product engineer. I had a chance to be responsible for the whole mobile architecture, the protocols, and API layers between apps, web and backend, the data storage and synchronization, backups and monitoring, etc.

Now I am working in a data security company[4], making software that protects data and prevents leaks, which is designed to be friendlier to developers who are not the from the "security planet", like my ex-colleagues. As I often say, "Cryptography should work everywhere," and so our open source libraries[5] and tools can be found in small mobile apps, as well as in large country-wide infrastructures. Every day I speak with other software companies who care about data security in their products, and I realized that they need help.

From these interactions, I noticed that most security people have no experience in developing decidedly usable software. On the other hand, most software developers don't have skills in security analysis and architecture. While I'm staying between two worlds, I feel this problem deeply.

Throwing Hot Potatoes

When I ask developers why they don't implement basic security-sanity features (like protecting user passwords, limiting access to user data, etc.), I often hear the same answer: "The manager didn't tell us to do it, we don't have this task on the board."

[2] https://www.researchgate.net/publication/303794257_What_is_the_value_of_security_Contextualising_the_negativepositive_debate

[3] https://globalnews.ca/news/4298279/hacker-hits-local-mattress-store-with-ransomware/

[4] https://www.cossacklabs.com/

[5] https://github.com/cossacklabs/acra

Imagine having the following conversations with managers again and again:

– How things are going with security in your app? – We are totally fine, we have smart and competent developers, they handle everything. – Does your team have security-related tasks? Do they assess the security risks while planning new features? Do you follow the Secure Development Life Cycle? Do you have an internal blue team? Do you do security audits once in a while? – Well... We don't do all of that.. but our QA team does pen-tests!

It's a great first step when developers try using pen-tests and security checklists. However, this is a low hanging fruit of "let's build and release quickly, developers will try to do their best, and a week before a release we will go through the OWASP checklist and solve the obvious problems." In the best-case scenario, the team will write down and solve the critical issues, but people usually only solve the easy things and put off the complicated ones to be implemented in the next releases – or, more typically, never.

Building secure software is hard. An attacker has to make just a few correct guesses, while a developer has to take care of a number of things from the very beginning.

Your system is not secure if you do SSL pinning and store keys in key storage in your app, but use an open MongoDB with default admin password on the backend. Data security works if it covers the system fully, including mobile and web applications, backends, external services, and backups. Solution architects and security team should care and align feature developers and devops. This approach is efficient when a company is not hiding under "we're fine" umbrella.

Secure Development

It is impossible to learn secure development just from reading tutorials.

Most "Building secure chat" tutorials starts with "Let's create a new project" and end up with the "Now we have encrypted data" step. Moreover, they use encryption libraries with APIs expecting developers to choose between symmetric cyphers, their mode *(ECB vs GCM, huh?)*, salt and "nonce," etc. Most developers just copy-paste cryptographic code snippets without understanding them. Even if data encryption is done correctly, one needs to design the rest of the application flow, use a different encryption for transferring the data, consider the key management procedures, choose appropriate authentication controls, techniques of monitoring and alerting, and so on.

As developers, we are always trying to cut a few corners, are not we?

I've recently discovered a "codeless" service that "protects" mobile applications you send it to them. So, imagine, you send a compiled .ipa or .apk and code signing credentials to their site, and they "magically" protect your app by integrating encryption, SSL pinning, DLP, and obfuscation. Surprisingly, neither managers nor developers are embarrassed. Close-source system that changes your application flow without providing you with a way to see changes, what could possibly go wrong?

The illusion of security is much worse than its absence.

Secure Software Development Lifecycle as a methodology that has existed for many years. It is described in detail as MS SDL[6] (deep and solid) and as OWASP S-SDLC[7] (short and

[6] http://www.microsoft.com/en-us/sdl

[7] http://www.owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project

modern). The SSDLC distils common sense from the industry experience of building secure software and prescribes techniques which cover most risks in most cases.

The SSDLC consists of several steps and accompanies a typical software development approach (including Agile and XP.)

- *Risk evaluation and assessment* – understanding business and technological risks threatening data of and on which the application operates.
- *Building a threat model* – what are the typical threats that the application faces?
- *Security roadmap* defined according to the most possible threats. Typically, a security roadmap contains data minimization, data protection during storage and transmission, access limitation and monitoring.
- *Secure coding* – which libraries and tools to use, where to store keys. Using a good encryption library itself won't make your app secure.
- *Secure operations* – infrastructure-level changes that should be done, including procedures of revoking user sessions, updating certificates, patching libraries.
- *Security verification and testing* are usually done combining manual and automation tools for continuously testing a code base, including its dependencies, for vulnerabilities.
- *Threat response and recovery* – the plan and procedures to conduct upon detecting threats.

The SSDLC is a continuous process that should start after suspecting any vulnerability or detecting an incident. And it's a process, not a single feature to add.

End-to-end Encryption And Marketing

How technically difficult you think it is to make end-to-end encrypted applications? Well, it's a bit tricky, but not impossible. Cryptography helps to reduce the attack surface and risks and prevents attackers and insiders from reading the data. If a system does not know which data it operates on, this data cannot be stolen easily, right?

So why do not we have more E2EE data exchange in modern software? If a company does not have access to its data, it cannot analyze it and use it for advertising. Finding the right balance between missed advertising profits and de-risking actual ownership of this data is tricky, but doable.

Tomorrow

Secure software development is far from being a popular practice, so we still need reminders to show us the real consequences of data security. Until good software and secure software become synonymous, there will be fines and losses. So, what exactly should we do tomorrow to improve the security of our software?

I don't have an easy answer for you. Maybe starting to devote time to security as a sign of professionalism – similarly to the way we focus on writing maintainable, testable, manageable code, not just code "that works"?

Cover photo by Nick Boyer[8] on Unsplash[9].

[8] https://unsplash.com/photos/96VU0vrYPtU?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

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

The Weakest Link

Adrian Kosmaczewski

December 3rd, 2018

At the end of 2000 I was studying IT management in the Universidad de Buenos Aires[1], and the teacher in front of my class was trying to instill some basic notions of computer security in the heads of a rather skeptical (albeit ignorant) crowd.

Out of a certain exasperation, said professor stopped the class. He asked one of the few lucky owners of a cellphone in the classroom for his phone number. After entering the number into a device looking like a handheld calculator, he asked this unsuspectful student to go outside the classroom and call a friend or a relative.

The student complied. After a few seconds, to our astonishment, the machine picked up the call. We could all clearly hear our friend talking to his mum about dinner plans for that evening.

I was shocked. I had no idea how cellphone networks worked. But I was under the absolute assumption that privacy and security were given attributes of them. That whoever designed and built that infrastructure, had them as major goals. Was not private correspondence protected by the constitutions of most countries in the Hemisphere, after all? Why would cellphone communications be any different? Was this assumption foolish?

[1] https://en.wikipedia.org/wiki/University_of_Buenos_Aires

Rude Awakening

A few months later, I bought a copy of "Hacking Exposed 2[2]" by Scambray, McClure and Kurtz; it was the only book about computer and Internet security I could find in the shelves of the stunning and recently opened "Ateneo Grand Splendid"[3] bookstore in Buenos Aires.

(It must also be said that this is one of the books that changed the meaning of the word "Hacking" forever in the minds of millions, a confusion that Richard Stallman himself has tried to debunk,[4] but to no avail.)

In that book I learnt that most of the traffic in our networks back then was completely unencrypted. Which meant that, with the proper setup, it was easily "sniffable."

First Steps

Out of curiosity, I used this brand new search engine you might have heard about, "Google," to learn more about the subject. I found a copy of CaptureNet, a freeware packet sniffer part of the SpyNet/PeepNet by Laurentiu Nicula[5]; then I looked up for the port number used by MSN Messenger[6] (it was 1863 in case you were wondering.) Finally I found out how to enable "promiscuous mode" in the network card in my laptop.

I plugged all of these pieces together in silence, my coworkers fully unaware of my proceedings. In the small LAN of the office my company had north of Buenos Aires, we were all using Windows 2000 Professional there. And, as it were, it turns out we were using a good old router, not a switch, which undoubtedly helped me fulfill this task.

I finally turned the sniffer on.

Instantaneously, my screen started to show me the conversations my peers were having on MSN Messenger. And I mean all of it. Every detail of their private lives, the current business deals, the comments of the latest news. Every "smiley" they were sharing. Everything in their private lives, every single word they said. All on my screen, ready to read, without any encryption.

After changing the sniffing port to 80, I used CaptureNet's uncanny feature of reconstructing the web pages. All of my colleagues browsing at that very moment, including images and scripts, appeared in my laptop.

Those sessions shall remain anonymous and forgotten. I got so scared that I basically hit the stop button on the sniffer and deleted the logs. For the first time in my career (I had been working as a software developer for 3 years so far) I had the sensation that everything we did in our industry was extremely fragile, insecure. It seemed to me that we were all blissfully unaware of how naked we were.

It was even worse than watching our teacher listening to private phone conversations. This was even simpler and cheaper – no need for custom hardware.

[2] https://www.amazon.com/Hacking-Exposed-Joel-Scambray-ebook/dp/B00UYDTBW6
[3] https://en.wikipedia.org/wiki/El_Ateneo_Grand_Splendid
[4] https://stallman.org/articles/on-hacking.html
[5] https://www.linkedin.com/in/mendark/
[6] https://en.wikipedia.org/wiki/Windows_Live_Messenger

Wannabe Cracker

This new knowledge took me to a rather somber hobby, one of which I am not really proud nowadays. Around 2002 I got into the habit of scanning random IP ranges on the Internet, finding computers running Windows 95 or 98 with port 139 (NetBIOS) wide open, and then connecting to them using Back Orifice[7].

Connected to those machines on the other side of the planet, I watched live. Those users were typing documents, filling spreadsheets, or browsing the web. I would then take over their mouse, open a Notepad file, paste a pre-written text explaining to them that I meant no harm. Explaining them that their system was open to intrusion, and how they should protect themselves.

I would then leave without further action, a shocking small Notepad window behind me. A mirror of their own nakedness, and most probably a gaze of terror in their eyes. In my defense I will say that I never, ever made any change or stole any information from those machines; my intent was to raise awareness, although I reckon my methodology was probably not the most adequate or tasteful.

Security breaches like these were by then already the matter of urban legend. Or worse, heated discussion around an almost expected feature in every new version of Windows. The situation of Microsoft regarding security was so serious that Bill Gates himself wrote the now legendary "Trustworthy Computing"[8] memo to his employees, making the book "Writing Secure Code"[9] by Howard and LeBlanc a mandatory reading inside his company.

The motto of Microsoft was "a computer in every desk." Unfortunately said motto made no mention of how secure that computer had to be. This is to me another proof that "moving fast and breaking things" is one of the most harmful policies a company can follow.

The Dawn Of A New Era

There was a bigger question that popped into my mind back then.

I was just a software developer without any kind of formal training, with just a basic amount of curiosity and a little experience. Yet I could put together all of those pieces in an admittedly standard computer. What could then be happening in governments or companies? What would be the level of intrusion of these companies in our lives? How much did our government and corporations know about us?

Well, in the case of the Argentinian government, not a lot. Its level of competence in IT was still a long way off (one could argue that ignorance was a bliss for the Argentinian people back then,) and there were other problems[10] to worry about.

But of course I figured out that great powers like the USA, Russia or Europe were easily reading (or at least storing) everything we said and did online. Because doing so was not only cheap for them (I had not spent a single dime in my setup, using a company laptop and some freely available software) but also extremely convenient and strategically important.

Actually, it made more sense for governments to actually record everything that was going online and storing it in a database, than not doing it at all. All things considered, having all of that information in a database for later evaluation was better than not having it.

[7] https://en.wikipedia.org/wiki/Back_Orifice
[8] https://www.wired.com/2002/01/bill-gates-trustworthy-computing/
[9] https://www.amazon.com/Writing-Secure-Second-Developer-Practices/dp/0735617228
[10] http://edition.cnn.com/2001/WORLD/americas/12/19/argentina.riots/

Turns out I was right, and yet, still very naive. I had not realized that nuclear plants, hospitals[11], pacemakers, finance and nearly everything that uses electricity, was already being managed with computers connected to the Internet.

"Geekonomics"[12] by David Rice would not be published before 2008. Bruce Schneier blog[13] was still just a newsletter[14]. And IoT was... well.

The World We Built

We, designers and users of the technology of the future, eager to use the latest gizmos and the newest of approaches, we were feeding a silent surveillance machine, the product of which, 18 years later, is the slow establishment of the largest coalition of fascist leaders in modern history.

As citizens, as technology designers and users, we all have contributed to the growth of this machine[15] and the birth of this new world order, through none other than Facebook, MSN Messenger, Skype, Twitter, Tumblr, iOS, Android, Google, and so many other companies and systems.

But we can change this, the same way we unconciously decided to make this happen.

And of all citizens, software engineers and IT experts have an ethical duty to make computers secure, acting in two fronts at once:

- First, by designing, implementing and deploying systems in a secure and privacy-conscious fashion.
- Second, by teaching and raising awareness of the various issues around security and privacy in our modern infrastructure.

Humans, and particularly those working with computers, are the weakest link in the chain of security.

The World We Could Have Built

We are the ones giving out our (weak) passwords when receiving a phone call from the "support" team of a company whose products we use, or when a stranger asks us for it on the street[16].

Clicking "I Agree" on most privacy agreements without even skimming the text.

Not (fully) understanding how encryption works, and why some algorithms are of no use today, and even better, rolling up[17] our own[18] without first reading Schneier's Memo to the Amateur Cipher Designer[19] written in... 1998.

Not even trying to configure PGP in our mail clients.

[11] https://abc.go.com/shows/greys-anatomy/episode-guide/season-14/8-winter-finale-out-of-nowhere
[12] https://www.amazon.com/Geekonomics-Real-Insecure-Software-paperback/dp/0321735978
[13] https://www.schneier.com/
[14] https://www.schneier.com/crypto-gram/
[15] https://www.bbc.com/ideas/videos/surveillance-capitalism-has-led-us-into-a-dystopia/p06p0tdy?playlist=imho
[16] https://www.linkedin.com/feed/update/urn:li:activity:6461866666249777152/
[17] https://security.stackexchange.com/q/25585
[18] https://security.stackexchange.com/q/18197
[19] https://www.schneier.com/crypto-gram/archives/1998/1015.html#cipherdesign

Designing purposedly complicated systems that others cannot properly configure, leaving security holes open.

Forgetting to add SSL certificates[20] to that new website we built for a relative.

Misconfiguring routers and firewalls for convenience... or plain ignorance.

Storing passwords as plain text in databases, and then, God forbid, sending them to users via e-mail. Not even trying to explain our pointy-haired bosses that this should not happen.

Ignoring two-factor authentication in our accounts.

Letting ourselves be easily spoofed by politicians, as they try to convince us how weakening encryption is important for our "security."

Our Duty

It is our duty, our ethical duty, the one with utmost importance, to realize that we are humans, and that we, the people, are the weakest link in the chain of security.

The good news is, this link, however weak, has an uncanny capacity to learn and change. We can change this world we built. We can stop building the current one, right away.

Let us build a world where we protect each other, consciously, all the time, and where the systems we build serve this purpose.

A world where privacy and security are basic human rights.

Let us teach each other how to build this new world, like my visionary teacher tried to.

Cover photo by Tan Kaninthanond[21] on Unsplash[22].

[20] https://letsencrypt.org/

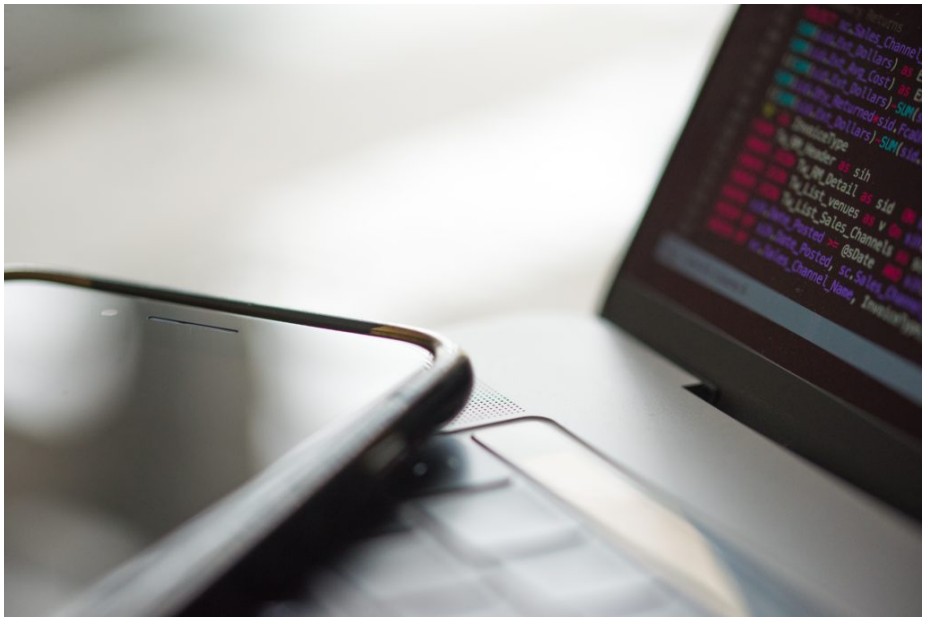
[21] https://unsplash.com/photos/VwxIW9BpAlA?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[22] https://unsplash.com/search/photos/spying?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

On Modern Security Culture

Graham Lee

December 3rd, 2018



Long before Adrian and I started *De Programmatica Ipsum*, we met at a conference in London where he had invited me to talk on mobile application security. Reflecting on this issue's topic, I compared that experience with the infosec-focussed events I have attended. I thought of the worst aspects of the infosec culture that I've seen. Writing about those would make for a full and interesting article, but wouldn't be particularly inspiring.

Instead, I'm going to talk about some of the more progressive parts of the infosec world, by introducing the people who informed my view on those parts. If they inspire you as much as they did me, then you will come to think of infosec as something your whole organisation, and your whole community, can come together on.

Window Snyder

Window Snyder[1] is currently Intel's Chief Software Security Officer. She has a history of transforming software security at computing's big names: Apple, Mozilla, and Microsoft are all former employers. At Microsoft, she was security lead for Windows XP Service Pack 2 and Windows 2003.

Trustworthy Computing

These products were released in the wake of Bill Gates' Trustworthy Computing memo[2].

[1] https://twitter.com/window

[2] https://www.itprotoday.com/strategy/complete-text-bill-gates-trustworthy-computing-memo

> Today, in the developed world, we do not worry about electricity and water services being available. With telephony, we rely both on its availability and its security for conducting highly confidential business transactions without worrying that information about who we call or what we say will be compromised. Computing falls well short of this, ranging from the individual user who isn't willing to add a new application because it might destabilize their system, to a corporation that moves slowly to embrace e-business because today's platforms don't make the grade.

After that memo, the company stopped development of its products to train development teams on software security and made security and trustworthiness top priorities.

Threat Modelling

Microsoft's approach to security during that time was shared widely. Their approach became the basis for Secure Development Lifecycles (SDLC) across the industry. Snyder's co-authorship with Frank Swiderski of Threat Modelling[3] is an important part of this outreach.

I believe this book shows how a modern, self-organising software product team can make itself security-infused. There's still a place for external consultants, penetration testers, and dedicated security teams. These are people who *inform* and *educate* the product team. The product team are ultimately aware of their product's security model, its risks and how to address those risks.

If your product team is not sure how to think about their product's security, get this book and run a workshop. If you and your team do not think security is your problem, then you *definitely* need to read and internalise the Threat Modelling book.

Kate Moussouris

k8em0[4] is also a Microsoft alumna. It's time to revise any opinion you have that Microsoft is not an innovative software engineering house. She has advanced the way that organisations including Microsoft and the U.S. Department of Defense interact with external security researchers and testers.

Moussouris was instrumental in introducing a "bug bounty" program at both organisations, and elsewhere. A bug bounty is a prize paid to an external reporter of a security vulnerability. Bug bounties are one incentive designed to support responsible disclosure, another of Moussouris's successes in the field.

Responsible Disclosure

The actors involved in security vulnerability discovery have different, conflicting objectives.

The product vendor may prefer never to admit that their software was insecure. They would like to silently patch their product (if they intend to patch it), and move on as if nothing had happened.

If an external security expert found the bug, they will want to publish details and get credit for the discovery. They may use that discovery in their marketing materials, as demonstration of their expertise.

[3] https://www.goodreads.com/book/show/628285.Threat_Modeling
[4] https://twitter.com/k8em0

Unfortunately another group who would benefit from public disclosure of security bugs is the attackers. They can use information about the bug to design an attack targeting people who haven't yet remediated the problem.

Which leads us to the customers, the people using the software. They want to know about the problems so that they can patch, or change their security policy, or otherwise address the risks associated with the vulnerability.

Responsible disclosure builds a compromise position from this conflict which represents the best trade-offs for all parties except the attackers. The person who discovered the flaw reports it privately to the vendor, and negotiates a timeline to public disclosure including full credit. The vendor gets time to fix the bug and get the fix to customers before public disclosure. Users get told about vulnerabilities, but only after the vendor has issued a patch.

Bug Bounties

Bug bounty programs act as a form of virtue signal for responsible disclosure. It's important to see that bounties are not a *replacement* for Threat Modelling and internal security practices. Instead, bug bounties indicate that a vendor acknowledges that external experts may find problems they have missed. A company with a bug bounty program is saying that it supports responsible disclosure, and will reward others who cooperate in its responsible disclosure approach.

The bounty is not paid to *anybody* who finds a security bug. Rather, the vendor pays those who engage in its responsible disclosure process. The vendor hopes that the value of the bounty payout stops the researcher from publicising a vulnerability as soon as they discover it. That situation is called a "zero day" or 0day, because attackers have the opportunity to exploit the problem on day zero of the disclosure and patch timeline, before the vendor has a chance to fix it.

Modern Security Culture

Through practices like threat modelling and responsible disclosure, information security has moved from a combative to a collaborative art. We no longer have a security team or an external penetration tester telling us "no" after we've built a product. We have devops and devsecops: a culture in which a software team works together and treats security like any other attribute of their product.

Such collaboration extends beyond the org chart, with multiple parties coordinating their vulnerability disclosures to the benefit of the people using our software. The work of Window Snyder, Kate Moussouris and others enables this cultural shift, helping our whole industry to make computers safer and more helpful.

Cover photo by Chris Ried[5] on Unsplash[6].

[5] https://unsplash.com/photos/M17OmBsh9xA?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

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