blog.cryptographyengineering.com

Was the Efail disclosure horribly screwed up? – A Few Thoughts on Cryptographic Engineering

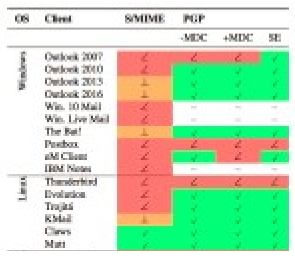
Matthew Green

13-16 minutes

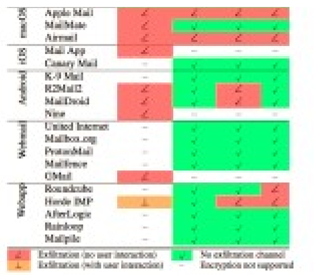
TL;DR. No. Or keep reading if you want.

On Monday a team of researchers from Münster, RUB and NXP disclosed serious cryptographic vulnerabilities in a number of encrypted email clients. The flaws, which go by the cute vulnerability name of "Efail", potentially allow an attacker to decrypt S/MIME or PGP-encrypted email with only minimal user interaction.

By the standards of cryptographic vulnerabilities, this is about as bad as things get. In short: if an attacker can intercept and alter an encrypted email — say, by sending you a new (altered) copy, or modifying a copy stored on your mail server — they can cause many GUI-based email clients to *send the full plaintext of the email to an attacker controlled-server*. Even worse, most of the basic problems that cause this flaw have been known for years, and yet remain in clients.



OS	Client	S/MIME	PGP		
			-MDC	+MDC	SE
Windows	Outlook 2007	∠	∠	∠	✓
	Outlook 2010	∠	✓	✓	✓
	Outlook 2013	⊥	✓	✓	✓
	Outlook 2016	⊥	✓	✓	✓
	Win. 10 Mail	∠	–	–	–
	Win. Live Mail	∠	–	–	–
	The Bat!	⊥	✓	✓	✓
	Postbox	∠	∠	∠	∠
	eM Client	∠	✓	∠	✓
	IBM Notes	∠	–	–	–
Linux	Thunderbird	∠	∠	∠	∠
	Evolution	∠	✓	✓	✓
	Trojitá	∠	✓	✓	✓
	KMail	⊥	✓	✓	✓
	Claws	✓	✓	✓	✓
	Mutt	✓	✓	✓	✓



macOS	Apple Mail	↯	↯	↯	↯
	MailMate	↯	✓	✓	✓
	Airmail	↯	↯	↯	↯
iOS	Mail App	↯	–	–	–
	Canary Mail	–	✓	✓	✓
Android	K-9 Mail	–	✓	✓	✓
	R2Mail2	↯	✓	↯	✓
	MailDroid	↯	✓	↯	✓
	Nine	↯	–	–	–
Webmail	United Internet	–	✓	✓	✓
	Mailbox.org	–	✓	✓	✓
	ProtonMail	–	✓	✓	✓
	Mailfence	–	✓	✓	✓
	GMail	↯	–	–	–
Webapp	Roundcube	–	✓	✓	↯
	Horde IMP	⊥	✓	↯	↯
	AfterLogic	–	✓	✓	✓
	Rainloop	–	✓	✓	✓
	Mailpile	–	✓	✓	✓

↯ Exfiltration (no user interaction) ✓ No exfiltration channel
⊥ Exfiltration (with user interaction) – Encryption not supported

Table 4: Exfiltration channels for various email clients for S/MIME, PGP SEIP with stripped MDC (*-MDC*), PGP SEIP with wrong MDC (*+MDC*), and PGP SE packets.

The big (and largely under-reported) story of EFail is the way it affects S/MIME. That "corporate" email protocol is simultaneously *(1)* hated by the general crypto community because it's awful and has a slash in its name, and yet *(2)* is probably the most widely-used email encryption protocol in the corporate world. The table at the right — excerpted from the paper — gives you a flavor of how Efail affects S/MIME clients. TL;DR it affects them very badly.

Efail *also* happens to affect a smaller, but non-trivial number of OpenPGP-compatible clients. As one might expect (if one has spent time around PGP-loving folks) the disclosure of these vulnerabilities has created something of a backlash on HN, and among people who make and love OpenPGP clients. Mostly for reasons that aren't very defensible.

So rather than write about fun things — like the creation of CFB and CBC gadgets — today, I'm going to write about something much less exciting: the problem of *vulnerability disclosure* in ecosystems like PGP. And how bad reactions to disclosure can hurt us all.

How Efail was disclosed to the PGP community

Putting together a comprehensive timeline of the Efail disclosure process would probably be a boring, time-intensive project. Fortunately Thomas Ptacek loves boring and time-intensive projects, and has already done this for us.

Briefly, the first Efail disclosures to vendors began *last October*, more than 200 days prior to the agreed publication date. The authors notified a large number of vulnerable PGP GUI clients, and also notified the GnuPG project (on which many of these projects depend) by February at the latest. From what I can tell every major vendor agreed to make some kind of patch. GnuPG decided that it wasn't their fault, and basically stopped corresponding.

All parties agreed not to publicly discuss the vulnerability until an agreed date in April, which was later pushed back to May 15. The researchers also notified the EFF and some journalists under embargo, but none of them leaked anything. On May 14 someone dumped the bug onto a mailing list. So the EFF posted a notice about the vulnerability (which we'll discuss a bit more below), and the researchers put up a website. That's pretty much the whole story.

There are three basic accusations going around about the Efail disclosure. They can be summarized as (1) maintaining embargoes in coordinated disclosures is really hard, (2) the EFF disclosure "unfairly" made this sound like a serious vulnerability "when it isn't", and (3) everything was already patched anyway so *what's the big deal*.

Disclosures are hard; particularly coordinated ones

I've been involved in two disclosures of flaws in open encryption protocols. (Both were TLS issues.) Each one poses an impossible dilemma. You need to simultaneously *(a)* make sure every vendor has as much advance notice as possible, so they can patch their software. But at the same time *(b) you need to avoid telling literally anyone,* because nothing on the Internet stays secret. At some point you'll notify some FOSS project that uses an open development mailing list or ticket server, and the whole problem will leak out into the open.

Disclosing bugs that affect PGP is particularly fraught. That's because there's no such thing as "PGP". What we have instead is a large and distributed community that revolves around the OpenPGP protocol. The pillar of this community is the GnuPG project, which maintains the core GnuPG tool and libraries that many clients rely on. Then there are a variety of niche GUI-based clients and email plugin projects. Finally, there are commercial vendors like Apple and Microsoft. (Who are mostly involved in the S/MIME side of things, and may reluctantly allow PGP plugins.)

Then, of course there are thousands of end-users, who will generally fail to update their software unless something really bad and newsworthy happens.

The obvious solution to the disclosure problem to use a staged disclosure. You notify the big commercial vendors first, since that's where most of the affected users are. Then you work your way down the "long tail" of open source projects, knowing that inevitably the embargo could break and everyone will have to patch in a hurry. And you keep in mind that *no matter what happens, everyone will blame you for screwing up the disclosure.*

For the PGP issues in Efail, the big client vendors are Mozilla (Thunderbird), Microsoft (Outlook) and maybe Apple (Mail). The very next obvious choice would be to patch the GnuPG tool so that it no longer spits out *unauthenticated* plaintext, which is the root of many of the problems in Efail.

The Efail team appears to have pursued exactly this approach for the client-side vulnerabilities. Sadly, the GnuPG team made the decision that it's not their job to pre-emptively address problems that they view as 'clients misusing the GnuPG API' (my paraphrase), even when that misuse appears to be rampant across many of the clients that use their tool. And so the most obvious fix for one part of the problem was not available.

This is probably the most unfortunate part of the Efail story, because in this case GnuPG is very much at fault. Their API does something that directly *violates* cryptographic best practices — namely, releasing unauthenticated plaintext prior to producing an error message. And while this could be understood as a reasonable API design at design time, continuing to support this API even as clients routinely misuse it has now led to flaws across the ecosystem. The refusal of GnuPG to take a leadership role in preemptively safeguarding these vulnerabilities both increases the difficulty of disclosing these flaws, and increases the probability of future issues.

So what went wrong with the Efail disclosure?

Despite what you may have heard, given the complexity of this disclosure, very little went wrong. The main issues people have raised seem to have to do with the contents of an EFF post. And with some really bad communications from Robert J. Hansen at the Enigmail (and GnuPG) project.

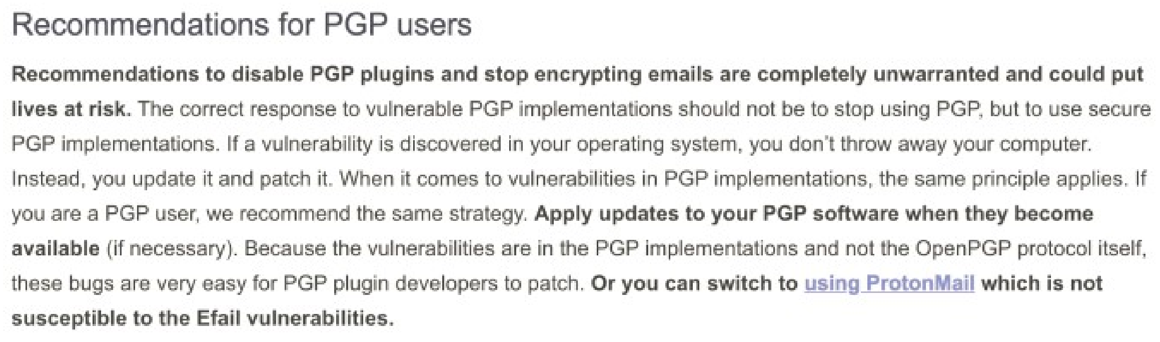
The EFF post. The Efail researchers chose to use the Electronic Frontier Foundation as their main source for announcing the existence of the vulnerability to the privacy community. This hardly seems unreasonable, because the EFF is generally considered a trusted broker, and speaks to the right community (at least here in the US).

The EFF post doesn't give many details, nor does it give a list of affected (or patched) clients. It does give two pretty mild recommendations:

1. Temporarily disable or uninstall your existing clients until you've checked that they're patched.
2. Maybe consider using a more modern cryptosystem like Signal, at least until you know that your PGP client is safe again.

This naturally led to a huge freakout by many in the PGP community. Some folks, including vendors, have misrepresented the EFF post as essentially pushing people to "permanently" uninstall PGP, which will "put lives at risk" because presumably these users (whose lives are at risk, remember) will immediately *fall back to sending incriminating information via plaintext emails* — rather than temporarily switching their communications to one of several modern, well-studied secure messengers, or just not emailing for a few hours.

In case you think I'm exaggerating about this, here's one reaction from ProtonMail:


Recommendations for PGP users

Recommendations to disable PGP plugins and stop encrypting emails are completely unwarranted and could put lives at risk. The correct response to vulnerable PGP implementations should not be to stop using PGP, but to use secure PGP implementations. If a vulnerability is discovered in your operating system, you don't throw away your computer. Instead, you update it and patch it. When it comes to vulnerabilities in PGP implementations, the same principle applies. If you are a PGP user, we recommend the same strategy. **Apply updates to your PGP software when they become available** (if necessary). Because the vulnerabilities are in the PGP implementations and not the OpenPGP protocol itself, these bugs are very easy for PGP plugin developers to patch. **Or you can switch to using ProtonMail which is not susceptible to the Efail vulnerabilities.**

The most reasonable criticism I've heard of the EFF post is that it doesn't give many details about which clients are patched, and which are vulnerable. This could presumably give someone the impression that this vulnerability is still present in their email client, and thus would cause them to feel less than secure in using it.

I have to be honest that *to me that sounds like a really good outcome.* The problem with Efail is that it doesn't matter if your client is secure. The Efail vulnerability could affect you if *even a single one of your communication partners* is using an insecure client.

So needless to say I'm not very sympathetic to the reaction around the EFF post. If you can't be sure whether your client is secure, you probably should feel insecure.

Bad communications from GnuPG and Enigmail. On the date of the disclosure, anyone looking for accurate information about security from two major projects — GnuPG and Enigmail — would not have been able to find it.

They wouldn't have found it because developers from *both Enigmail and GnuPG* were on mailing lists and Twitter claiming that they had never heard of Efail, and hadn't been notified by the researchers. Needless to say, these allegations took off around the Internet, sometimes *in place* of real information that could have helped users (like, whether either project had patched.)

It goes without saying that neither allegation was actually true. In fact, both project members soon checked with their fellow developers (and their memories) and found out that they'd both been given months of notice by the researchers, and that Enigmail had even developed a patch. (However, it turned out that even this patch may not perfectly address the issue, and the community is still working to figure out exactly what still needs to be done.)

This is an understandable mistake, perhaps. But it sure is a bad one.

PGP is bad technology and it's making a bad community

Now that I've made it clear that neither the researchers nor the EFF is out to get the PGP community, let me put on my mask and horns and tell you why *someone should be.*

I've written extensively about PGP on this blog, but in the past I've written mostly from a technical point of view about the problems with PGP. But what's really problematic about PGP is not just the cryptography; it's the story it tells about path dependence and how software communities work.

The fact of the matter is that OpenPGP is not really a cryptography project. That is, it's not held together by cryptography. It's held together by backwards-compatibility and (increasingly) a kind of an obsession with the idea of PGP as an end in and of itself, rather than as a means to *actually make end-users more secure.*

Let's face it, as a protocol, PGP/OpenPGP is just not what we'd develop if we started over today. It was formed over the years out of mostly experimental parts, which were in turn replaced, bandaged and repaired — and then worked into numerous implementations, which all had to be insanely flexible and yet compatible with one another. The result is bad, and most of the software implementing it is worse. It's the equivalent of a beloved antique sports car, where the electrical system is totally shot, but it still drives. You know, the kind of car where the owner has to install a hand-switch so he can turn the reverse lights on manually whenever he wants to pull out of a parking space.

If PGP went away, I estimate it would take the security community less than a year to entirely replace (the key bits of) the standard with something much better and modern. It would have modern crypto and authentication, and maybe even extensions for future post-quantum future security. It would be *simple*. Many bright new people would get involved to help write the inevitable Rust, Go and Javascript clients and libraries.

Unfortunately for us all, (Open)PGP does exist. And that means that even fancy greenfield email projects feel like they need to support OpenPGP, or at least some subset of it. This in turn perpetuates the PGP myth, and causes other clients to use it. And as a direct result, even if some clients re-implement OpenPGP from scratch, other clients will end up using tools like GnuPG which will support unauthenticated encryption with bad APIs. And the cycle will go round and around, like a spaceship stuck near the event horizon of a black hole.

And as the standard perpetuates itself, largely for the sake of being a standard, it will fail to attract new security people. It will turn away exactly the type of people who should be working on these tools. Those people will go off and build encryption systems in a totally different area, or they'll get into cryptocurrency. And — *with some exceptions* — the people who work in the community will increasingly work in that community because they're supporting PGP, and not because they're trying to seek out the best security technologies for their users. And the serious (email) users of PGP will be using it because they like the *idea of using* PGP better than they like using an actual, secure email standard.

And as things get worse, and fail to develop, people who work on it will become more dogmatic about its importance, because it's something threatened and not a real security protocol that anyone's using. To me that's where PGP is going today, and that is why the community has such a hard time motivating itself to take these vulnerabilities seriously, and instead reacts defensively.

Maybe that's a random, depressing way to end a post. But that's the story I see in OpenPGP. And it makes me really sad.