That Fun Time I Made A New OpenPGP Keypair, Got My Public Key Signed By Richard Stallman Himself, Deleted My Only Copy Of The Private Key The Next Day, And Used Crude Forensics Techniques To Recover The Lost Key Material Four Days Later

by **Georgiy Treyvus**

$ whoami

+ technical generalist with strong interests in math, programming, data analysis, and (cyber)security

+ specialist in web automation, data collection, reconnaissance, OSINT, OPSEC, counterintelligence, and offensive data analysis

HOTEL PENNSYLVANIA
HOTEL PENNSYLVANIA
MACY'S
sbarro the Italian
Tim Hortons
coffee & bake shop

Why OpenPGP These Days?

+ it’s still very widely used for code signing

+ it’s still the least worst option on fronts like having a standardized, importable, and exportable key format so I can freely reinstall my OS

+ theoretically app agnostic

Why A New Keypair?

+ why not?

+ **--enable-large-rsa** finally worked without Fedora’s version of GnuPG complaining about being compiled without **--enable-large-secmem**

Why A New Keypair?

+ significant gap between the strengths of the public key and symmetric ciphers

+ though not strictly necessary after CVE-2016-6313 it certainly wouldn’t hurt either and my last key was made just before the issue was fixed

Why A New Keypair?

+ general paranoia

+ this one is differently structured to work around a bug in GnuPG

+ actually one of several keypairs though we’ll only be focusing on one of them today

What Bug Is This?

+ it’s not possible to protect the private key material of the master key and the various subkeys with different passphrases

+ attempting to change the passphrase for a specific subkey or set of subkeys changed it for all subkeys

Ramifications

+ this severely limits one’s ability to compartment, and do damage control in case of compromise

+ happily some benefits of having separate subkeys still remain such as safety from certain classes of cross protocol attacks

My Workaround

```
= long term identity key
  = keysigning key
    = casual encryption
      and signing key(s)
    = longish term message
      signing key(s)
    = isolated/compartmented
      key(s) for operations
      where pseudonymity is
      not needed or wanted
```

About My Threat Model

+ What if I get compromised and my attacker then has a way to log keystrokes, read my files, or even arbitrary bytes of disk or memory?

+ Given my OpenPGP usage patterns what can I do to minimize the impact of such a compromise?

About My Threat Model

+ I decrypt messages often.

+ I sign messages rarely contra the bad advice of most OpenPGP tutorials. I’m generally fine without proof I said whatever dumb thing.

+ I sign keys probably even more rarely than messages.

About My Threat Model

+ To compromise a (sub)key the attacker needs either the raw private key material exposed in memory **or** the encrypted private key file contents and passphrase.

+ Because I decrypt often my encryption subkey would get stolen relatively instantly.

About My Threat Model

+ I can go weeks/months without signing messages or keys so stealing those (sub)keys will take time.

+ If I later detect a compromise by some means likely it will only be partial and I can then take steps to recover.

About My Threat Model

+ For instance I can scrub my compromised machine for some appropriate definition of scrub.

+ Alternatively I can work with the backed up key material on a totally separate and hopefully not compromised computer.

About My Threat Model

+ Ultimately once I’m at a safe computer I can for example revoke and replace a compromised encryption (sub)key.

+ My signing subkey and master key would still be safe enough.

About My Threat Model

+ Yes my attacker has an encrypted copy of my other private key material they didn't yet steal.

+ Still I'm not too worried.

+ I can choose brutishly strong passphrases with 40+ well diversified characters.

About My Threat Model

+ Now these passphrases aren’t truly random due to my poor (even for a human) memory.

+ Still I believe I can fake randomness well enough that I’m not too concerned.

About My Threat Model

The ultimate goal here is to be able to potentially recover from a reasonably significant compromise without having to generate totally new keys and do the whole fingerprint verification dance with all my cryptobuddies over the phone, in person, etc.

About My Threat Model

+ My threat model is predicated on me being able to discover that I am compromised.

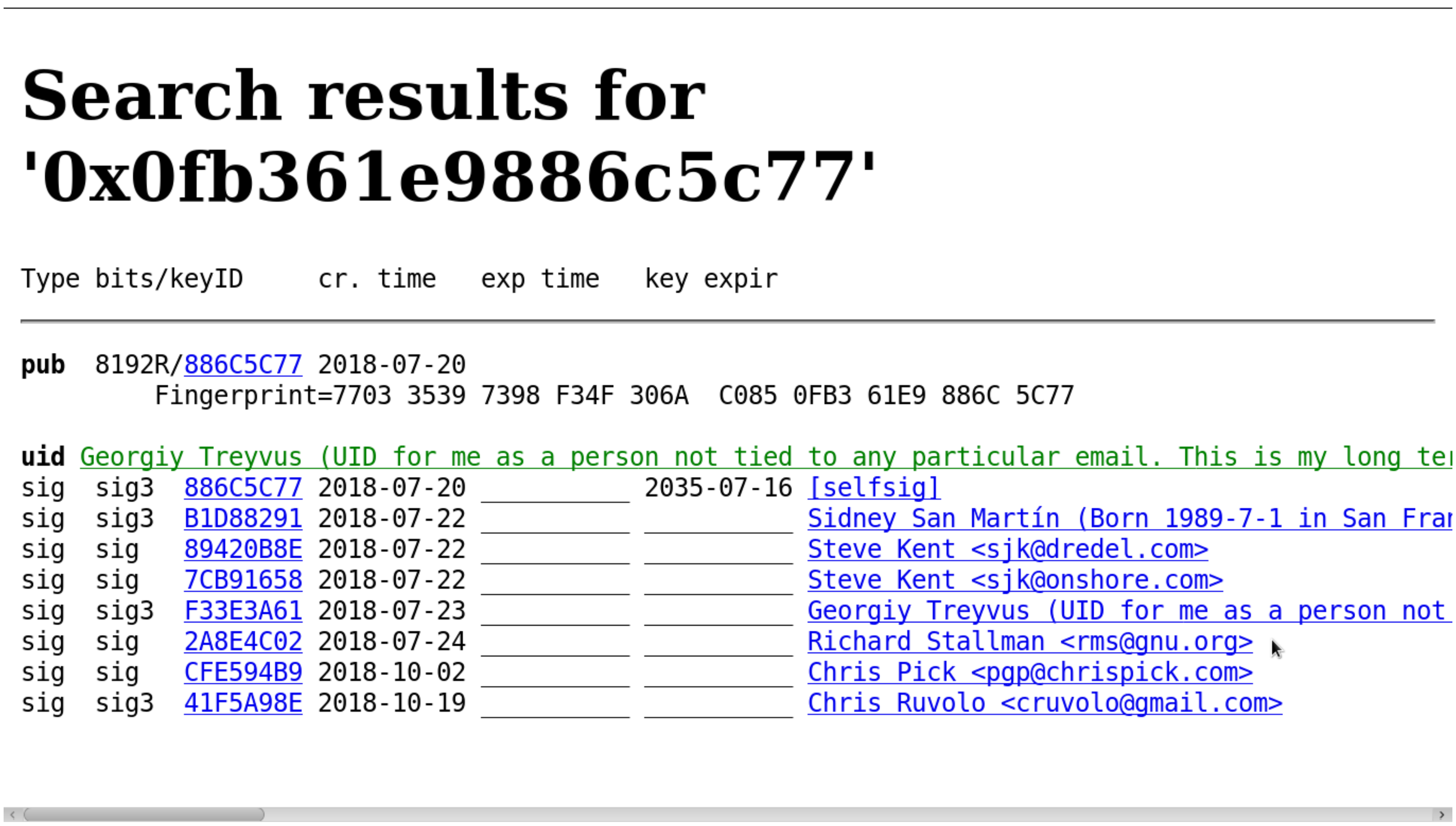
+ Furthermore it assumes that I can reliably put an upper bound on how long ago the compromise occurred.

About My Threat Model

+ assumes ability to remember or determine last time given subkeys were used

+ not a perfect or even realistic threat model in many senses

+ still compares relatively well with others I’ve heard



Search results for '0x0fb361e9886c5c77'

```
Type bits/keyID     cr. time   exp time   key expir

pub  8192R/886C5C77 2018-07-20
         Fingerprint=7703 3539 7398 F34F 306A  C085 0FB3 61E9 886C 5C77

uid Georgiy Treyvus (UID for me as a person not tied to any particular email. This is my long ter
sig  sig3  886C5C77 2018-07-20 __________ 2035-07-16 [selfsig]
sig  sig3  B1D88291 2018-07-22 __________ __________ Sidney San Martín (Born 1989-7-1 in San Frar
sig  sig   89420B8E 2018-07-22 __________ __________ Steve Kent <sjk@dredel.com>
sig  sig   7CB91658 2018-07-22 __________ __________ Steve Kent <sjk@onshore.com>
sig  sig3  F33E3A61 2018-07-23 __________ __________ Georgiy Treyvus (UID for me as a person not
sig  sig   2A8E4C02 2018-07-24 __________ __________ Richard Stallman <rms@gnu.org>
sig  sig   CFE594B9 2018-10-02 __________ __________ Chris Pick <pgp@chrispick.com>
sig  sig3  41F5A98E 2018-10-19 __________ __________ Chris Ruvolo <cruvolo@gmail.com>
```

Richard Stallman's personal site.

https://stallman.org

For current political commentary, see the daily political notes.

RMS' Bio | The GNU Project

I'm told that key servers carry many phony keys claiming to be mine. Here is info about which keys are really mine.

Old key (don't use it nowadays)

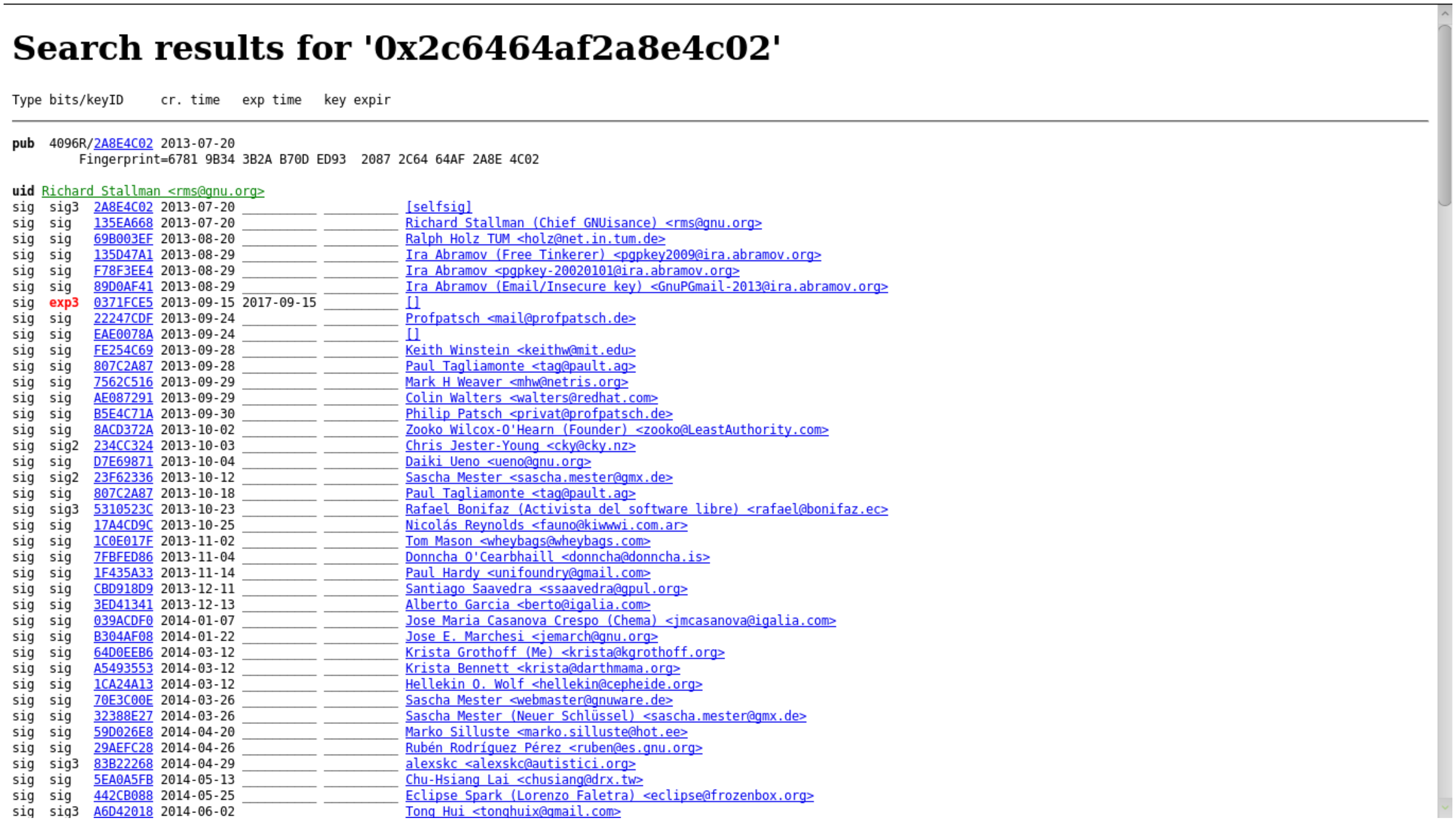
```
pub   1024D/135EA668 2001-03-05
uid                  Richard Stallman (Chief GNUisance) <rms@gnu.org>
sub   1024g/B1B10ED6 2001-03-05
```

New key

```
pub   4096R/2C6464AF2A8E4C02 2013-07-20
      Key fingerprint = 6781 9B34 3B2A B70D ED93  2087 2C64 64AF 2A8E 4C02
uid                  Richard Stallman <rms@gnu.org>
sub   4096R/2F30A2E162853425 2013-07-20
```

Of course, to be really sure which key is mine, you need to get my key fingerprint from me or follow a chain of signatures. If a phony key appears to be signed by someone you trust, you should see what's up with that person.

If you want an encrypted response, you must send me your key, because I don't use key servers. I don't promise to keep it permanently if we don't talk often, so if you talk with me again a year later you should send it again.



Search results for '0x2c6464af2a8e4c02'

```
Type bits/keyID     cr. time   exp time   key expir

pub  4096R/2A8E4C02 2013-07-20
         Fingerprint=6781 9B34 3B2A B70D ED93  2087 2C64 64AF 2A8E 4C02

uid Richard Stallman <rms@gnu.org>
sig  sig3  2A8E4C02 2013-07-20 __________ __________ [selfsig]
sig  sig   135EA668 2013-07-20 __________ __________ Richard Stallman (Chief GNUisance) <rms@gnu.org>
sig  sig   69B003EF 2013-08-20 __________ __________ Ralph Holz TUM <holz@net.in.tum.de>
sig  sig   135D47A1 2013-08-29 __________ __________ Ira Abramov (Free Tinkerer) <pgpkey2009@ira.abramov.org>
sig  sig   F78F3EE4 2013-08-29 __________ __________ Ira Abramov <pgpkey-20020101@ira.abramov.org>
sig  sig   89D0AF41 2013-08-29 __________ __________ Ira Abramov (Email/Insecure key) <GnuPGmail-2013@ira.abramov.org>
sig  exp3  0371FCE5 2013-09-15 2017-09-15 __________ []
sig  sig   22247CDF 2013-09-24 __________ __________ Profpatsch <mail@profpatsch.de>
sig  sig   EAE0078A 2013-09-24 __________ __________ []
sig  sig   FE254C69 2013-09-28 __________ __________ Keith Winstein <keithw@mit.edu>
sig  sig   807C2A87 2013-09-28 __________ __________ Paul Tagliamonte <tag@pault.ag>
sig  sig   7562C516 2013-09-29 __________ __________ Mark H Weaver <mhw@netris.org>
sig  sig   AE087291 2013-09-29 __________ __________ Colin Walters <walters@redhat.com>
sig  sig   B5E4C71A 2013-09-30 __________ __________ Philip Patsch <privat@profpatsch.de>
sig  sig   8ACD372A 2013-10-02 __________ __________ Zooko Wilcox-O'Hearn (Founder) <zooko@LeastAuthority.com>
sig  sig2  234CC324 2013-10-03 __________ __________ Chris Jester-Young <cky@cky.nz>
sig  sig   D7E69871 2013-10-04 __________ __________ Daiki Ueno <ueno@gnu.org>
sig  sig2  23F62336 2013-10-12 __________ __________ Sascha Mester <sascha.mester@gmx.de>
sig  sig   807C2A87 2013-10-18 __________ __________ Paul Tagliamonte <tag@pault.ag>
sig  sig3  5310523C 2013-10-23 __________ __________ Rafael Bonifaz (Activista del software libre) <rafael@bonifaz.ec>
sig  sig   17A4CD9C 2013-10-25 __________ __________ Nicolás Reynolds <fauno@kiwwwi.com.ar>
sig  sig   1C0E017F 2013-11-02 __________ __________ Tom Mason <wheybags@wheybags.com>
sig  sig   7FBFED86 2013-11-04 __________ __________ Donncha O'Cearbhaill <donncha@donncha.is>
sig  sig   1F435A33 2013-11-14 __________ __________ Paul Hardy <unifoundry@gmail.com>
sig  sig   CBD918D9 2013-12-11 __________ __________ Santiago Saavedra <ssaavedra@gpul.org>
sig  sig   3ED41341 2013-12-13 __________ __________ Alberto Garcia <berto@igalia.com>
sig  sig   039ACDF0 2014-01-07 __________ __________ Jose Maria Casanova Crespo (Chema) <jmcasanova@igalia.com>
sig  sig   B304AF08 2014-01-22 __________ __________ Jose E. Marchesi <jemarch@gnu.org>
sig  sig   64D0EEB6 2014-03-12 __________ __________ Krista Grothoff (Me) <krista@kgrothoff.org>
sig  sig   A5493553 2014-03-12 __________ __________ Krista Bennett <krista@darthmama.org>
sig  sig   1CA24A13 2014-03-12 __________ __________ Hellekin O. Wolf <hellekin@cepheide.org>
sig  sig   70E3C00E 2014-03-26 __________ __________ Sascha Mester <webmaster@gnuware.de>
sig  sig   32388E27 2014-03-26 __________ __________ Sascha Mester (Neuer Schlüssel) <sascha.mester@gmx.de>
sig  sig   59D026E8 2014-04-20 __________ __________ Marko Silluste <marko.silluste@hot.ee>
sig  sig   29AEFC28 2014-04-26 __________ __________ Rubén Rodríguez Pérez <ruben@es.gnu.org>
sig  sig3  83B22268 2014-04-29 __________ __________ alexskc <alexskc@autistici.org>
sig  sig   5EA0A5FB 2014-05-13 __________ __________ Chu-Hsiang Lai <chusiang@drx.tw>
sig  sig   442CB088 2014-05-25 __________ __________ Eclipse Spark (Lorenzo Faletra) <eclipse@frozenbox.org>
sig  sig3  A6D42018 2014-06-02                       Tong Hui <tonghuix@gmail.com>
```

Let’s Open Pandora’s Box...

+ having made a new keypair in a hurry it was well past time to actually back it up especially in light of the new developments

+ but first some tests to make sure all was in order and further secure my setup so I back up a good state...

Why Test? Don't I Trust GPG?

+ not particularly

+ it's hardly the only tool with a poor, confusing, and brain hostile user interface but...

+ there's a **VERY LARGE** gap between GnuPG's perceived and actual state/behavior

Why Test? Don't I Trust GPG?

Pop Quiz! Which of the below keys is stronger?

```
pub   rsa8192 2018-10-25 [SCEA] [expires: 2019-10-25]
      2515BFC828FFEF899722F72804DC88CF8647510A
uid           [ultimate] Kubra Balik
```

```
pub   rsa8192 2018-10-25 [SCEA] [expires: 2019-10-25]
      B0E955E53E9818B3D8CC17AF8E296A711A5FB808
uid           [ultimate] Aydin Bayat
```

(They may look the same but I promise you they're not.)

Upon Taking A Closer Look

```
gpg> showpref
[ultimate] (1). Kubra Balik
     Cipher: AES256, AES192, AES, 3DES
     Digest: SHA512, SHA384, SHA256, SHA224, SHA1
     Compression: ZLIB, BZIP2, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify


gpg> showpref
[ultimate] (1). Aydin Bayat
     Cipher: 3DES
     Digest: SHA1
     Compression: ZIP, Uncompressed
     Features: MDC, Keyserver no-modify
```

Gap of 80+ bits of security!

Hence My Tests Involved

+ seeing if I could import and export my various keys

+ creating, certifying, and verifying signatures on various dummy keys

+ removing ~/.gnupg between rounds of tests for a clean slate which had consequences

Thickening The Plot A Bit

+ I’d inadvertently deleted ~/.gnupg/openpgp-revocs.d

+ ordinarily this would be fine as I usually manage my revocation certificates myself by other means which are more robust to things like OS reinstalls

Thickening The Plot A Bit

+ however in this case it wasn’t

+ I didn’t generate seperate revocation certificates due to the hurry I was in making my keys before HOPE

+ hence leaving me no way to revoke the soon deleted key

How I Deleted My Private Key

+ I was trying to harden my setup to what I felt was the optimal state before backing up my keys.

+ With OpenPGP private key data is usually protected using some symmetric algorithm whose key is derived from the passphrase

How I Deleted My Private Key

+ I wanted to protect the key material with 256 bit Camellia which I felt was the best symmetric cipher OpenPGP supported and felt it to be safer than the default of 128 bit AES.

+ also to optimize the other key derivation parameters

How I Deleted My Private Key

+ hence I ran **--export-secret-keys** with the strictest **--s2k-*** parameters

+ **HOWEVER** the private key material for my long term identity key wasn’t in GnuPG’s working keyring at that moment.

How I Deleted My Private Key

+ Obviously GnuPG couldn’t export private key material it didn’t have.

+ It should have halted immediately and complained that it couldn’t export what it didn’t have.

How I Deleted My Private Key

+ instead GnuPG deleted my only copy of the private key file which I was trying to overwrite via **--output**

+ given my threat model exporting to a new file, then using **shred** on the old would have been safer but I was kind of lazy here :-(

How I Deleted My Private Key

+ yes I was shown a warning about overwriting it

+ still that warning didn’t concern me much and I’d said yes to it before

+ for example when (re)exporting my private key after changing my passphrase

How I Deleted My Private Key

+ hence I entered y at the prompt as it was totally safe to do based on all my previous experience

+ this time was different

+ after I ran the export GPG warned that no private key material had been exported

How I Deleted My Private Key

WELL

THAT

SURE

WAS

OMINOUS

How I Deleted My Private Key

+ So I decided to check on it and sure enough my only copy of the private key was gone!

+ I was pretty sure I didn’t have a chance to back up what I had but checked my external hard drive anyway. No luck.

Depression Set In...

+ and hoo boy, did it really make itself comfortable...

+ I thought I was doomed.

+ I began writing a postmortem about my mistakes and the insane behavior of GnuPG that I wanted to post to my Diaspora profile.

What Now?

+ I had no cryptographic means to revoke the key but thought of weakening its cryptographic standing via social means

+ for example by contacting signatories and asking them to revoke their signatures

But...

+ admitting defeat publicly was embarrassing

+ and allegedly forensics people recover deleted data all the time...

+ so why not try?

+ rumor is I’m technical...

So Next

+ I unmounted my data volume which held the deleted key file, something which I should frankly have done immediately.

+ Sadly it was only a few hours after the incident that I started thinking clearly enough to do that.

+ I was **extremely lucky** here because I had a data partition instead of a partition for /home which is the more traditional setup one usually sees.

+ Were my setup more traditional for all I know the private key data would have been overwritten by say Firefox caching favicons...

Failed Recovery Attempt 1

+ after some research I learned about extundelete

+ which would have done exactly what I needed

+ if only it didn’t segfault every time I ran it :-(

Failed Recovery Attempt 1

+ What if the size of some system specific data structure changed or somehow differed from what Fedora’s packaged version assumed?

+ Will extundelete work if I recompile it?

+ I couldn’t recompile it.

Failed Recovery Attempt 1

+ I tried to get to the bottom of why and fix it

+ ultimately gave up after wrestling with the code for a few hours

Failed Recovery Attempt 2

+ I reached out to my friend Jay Michael Roberts a computer forensics expert.

+ He recommended CAINE (Computer Aided Investigative Environment) Live DVD

+ So I gave it a spin...

Failed Recovery Attempt 2

+ There’s loads of functionality there but how to discover what I need?

+ one of the third party manuals mentioned on their site introduced me to TSK (The Sleuth Kit)

Failed Recovery Attempt 2

+ Had the deleted key file been on an ext2 filesystem a combination of istat, and icat commands would work.

+ Sadly this was ext4 which works differently and clears direct block data from inodes upon file deletion.

Enter The Great Epiphany...

+ Unfortunately ext4 is not friendly to file recovery using filesystem metadata.

+ But I realized **I was dealing with data that had some well structured parts**.

+ This could be used to great advantage... :-) :-D

Private Key File Structure

Usually they're exported in armored ASCII format so:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
<private key material here>
-----END PGP PRIVATE KEY BLOCK-----
```

And odds are good my 8K keyfile was small enough to be stored contiguously...

SO PEANUT BUTTER REGEX TIME?

Maybe...

Problems

+ I’m well versed in the theory behind regular languages, finite automata, and all that good stuff.

+ I use regexes in my programs on rare occasions.

+ But I’m not gonna lie, I’m not that good at them.

Problems

+ My main problem with regexes is that they are cryptic and hard to read.

+ Maybe I’d use them more often given a friendlier interface in the form of tested, robust, mature, and widespread combinator libraries or something...

Problems

+ As the joke goes a senior programmer is one who knows how to use regexes but has the wisdom not to.

+ As JWZ said you have a problem, you decide to use regexes, then you have two problems.

Problems

+ this would require a very convoluted regex at the baseline

+ compounded further by the fact that after the dashed begin line there could be various optional headers that could contain almost any characters.

Problems

+ is whatever regex engine I’d be using in single line mode or multiline mode?

+ null characters/bytes could be problematic to some tools/libraries

+ many tools aren’t built for arbitrary binary data

Problems

+ lots of libraries/tools take a string as input and have no easy, obvious, or sane way to stream the 450G of my data volume to them as input

+ regexes are greedy by default

Insane Solutions Considered

+ I needed to chug through a lot of data fast so I thought maybe to use C++'s new <regex> library

+ upon looking up the docs my computer shut down to avoid permanent damage

+ file I/O is bottleneck

The Real Hero Of This Story

+ is without a shadow of a doubt my friend **Paul Backus**

+ without him this story would quite likely have a very different ending

The Real Hero Of This Story

+ Paul is an absolute systems programming powerhouse well on his way to being the next Andrei Alexandrescu or Graydon Hoare.

+ I exaggerate only slighty when I say he taught me all I know on good engineering.

The Real Hero Of This Story

+ he helped me stay calm

+ he was a great person to bounce various ideas off and unlike me was/is both rational and clearheaded

+ we considered various solutions from regexes to custom retrieval programs

The Real Hero Of This Story

+ It was Paul who came up with the solution that ultimately worked...

+ and boy was it **AWK**ward

+ so it was time to **release the krAWKen!**

+ but not my other bad jokes

The Solution That Worked

```
+ cat
/dev/mapper/logical_volumes-
data_volume | LC_ALL=C awk
'/-----BEGIN PGP PRIVATE KEY
BLOCK-----/,/-----END PGP
PRIVATE KEY BLOCK-----/' > /
home/georgiy/RECOVERED_STUFF
_LC_ALL_SET_TO_C
```

+ then postprocess with Vim

The Rocky Home Straightaway

+ passing the raw device file corresponding to my data volume as an argument to awk in the normal way didn’t work for some reason

+ awk couldn’t open the device file even though I was root, I had to stream via cat as a workaround

The Rocky Home Straightaway

+ Next came the finicky process of manually removing the binary cruft at beginning of the key block start lines from a defensive copy of the recovered data.

+ So I dove in with Vim and then fed the result of that to **gpg2 --import**

The Rocky Home Straightaway

+ GnuPG wasn’t able to parse out the key material of the lost key

+ I was disappointed and again was within a hair of giving up

The Rocky Home Straightaway

+ but I’m very stubborn

+ and knew with Vim it’s **very** easy to screw up

+ also even if I somehow don’t screw up Vim still might

The Rocky Home Straightaway

```
:for x in [1,2,3,4]
:  echo x
1
:  endfor
2
3
4
```

Need I say more?

The Rocky Home Straightaway

I’ve honestly lost count of how often I forgot I was in normal mode instead of insert, typed some keys, found a (subtly) garbled screen and then didn’t even know how many times to press **u** to return to a good state as I didn’t quite remember what it even looked like.

The Rocky Home Straightaway

+ In light of how finicky the whole cleaning process was and how reliably I make mistakes I thought I’d try again and be more careful.

+ To this day I’m not entirely sure what I did right this time around but the import worked.

VICTORY!!!!!!!!!!!!!!!!!!!!!

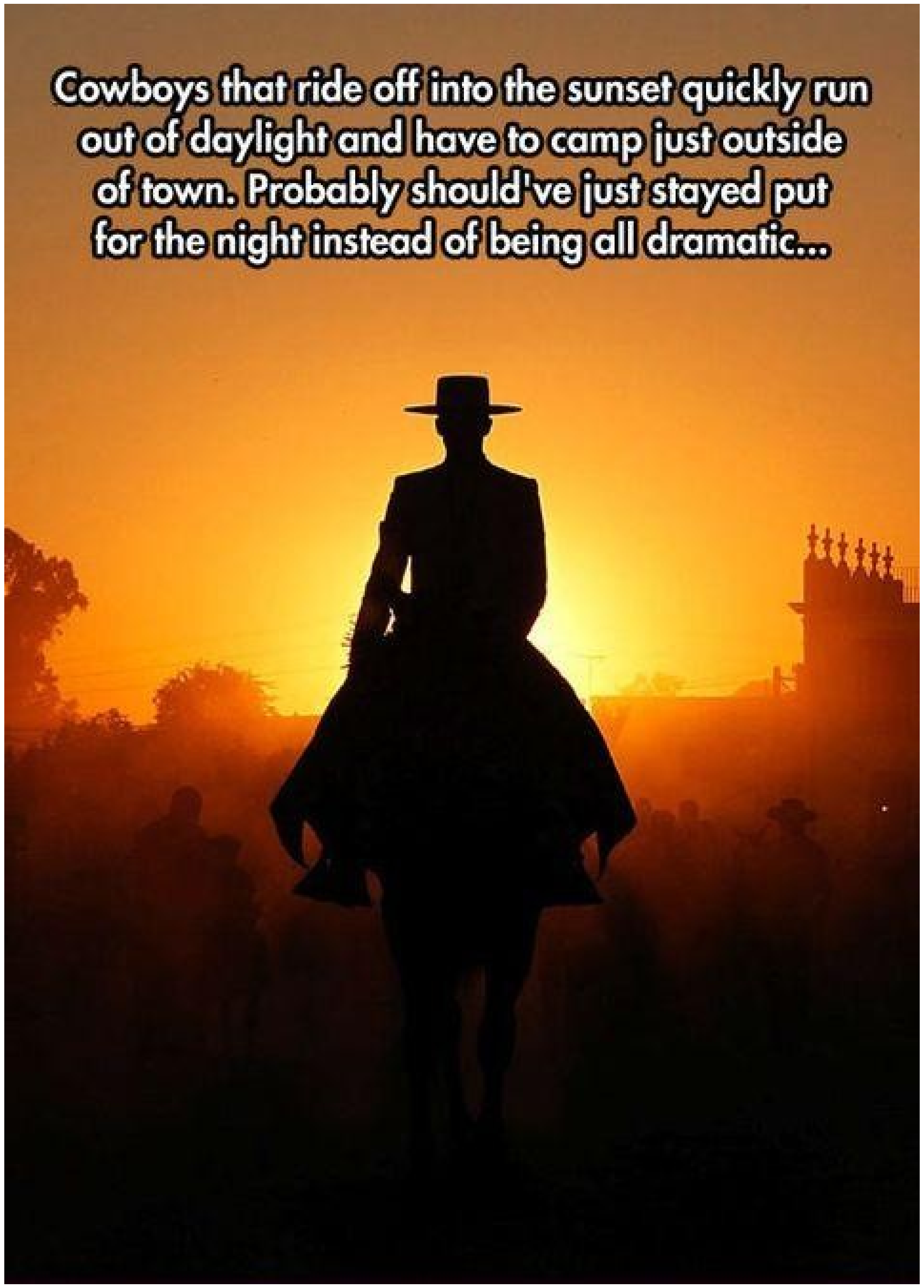
+ I can’t even begin describing my feeling when prompted for my long term identity key’s passphrase.

+ Further testing that involved signing/certifying and verifying the signatures on dummy keys confirmed the recovery was successful!

VICTORY!!!!!!!!!!!!!!!!!!!!!

+ Thus concludes the story of my first file carving adventure as this technique is known in the world of digital forensics.

+ I’m hoping to do a lot more of it out of curiosity and not necessity in the future.


Cowboys that ride off into the sunset quickly run out of daylight and have to camp just outside of town. Probably should've just stayed put for the night instead of being all dramatic...

Alternative Approaches

+ At the time I didn’t know about grep’s -a and -b options. Combining them would have given me interesting byte offsets to try inspecting in say Vim and then copy pasting...

+ grep -ao may also have worked aside from greediness

Alternative Approaches

+ maybe that could also be worked around with GNU grep’s experimental -P option

+ or not as grep is line based...

Best Alternative Approach

In retrospect a custom program to read 32K from every offset with the key block start characters to temp files. Sure this may have pulled in some cruft at the end but that doesn’t matter. --import will still successfully parse the result before crashing.

Takeaways

+ Sometimes shockingly dumb ramshackle techniques work better than the smart/professional ones.

+ Back up important stuff as soon as possible even if it’s not in the ideal state. Yes GnuPG was stupid but I was stupider. Period.

Takeaways

+ OpenPGP has hope if they update their specs to get rid of historical cruft.

+ GnuPG (and the garbage built on top of it) is utterly doomed.

+ don’t even get me started on pEp...

(Hot) Takeaways

+ I’m not necessarily saying to roll your own crypto but given how low the bar is it’s getting pretty hard to see how we could do worse.

+ don’t get cocky, but do question the crypto elitism

+ seriously get involved

(Hot) Takeaways

+ This fiasco, heartbleed, the libssh bug, etc, are all indicative of an inability to handle basic business logic let alone correctly reason about cache timing side channels or whatever. And that usability thing...

+ We need a new guard.

(Hot) Takeaways

+ Maybe it’s already here. OpenBSD’s built and deployed signify to help assure the integrity of their releases. The sky hasn’t fallen.

+ Ted Unangst’s reop also looks like a promising alternative solution in OpenPGP’s problem space

Thanks To:

Jay Michael Roberts

MASSIVE Thanks To:

Paul Backus

QED

+ questions

+ comments

+ book/movie deals

+ **consulting opportunities**