Havarirapport: Heartbleed

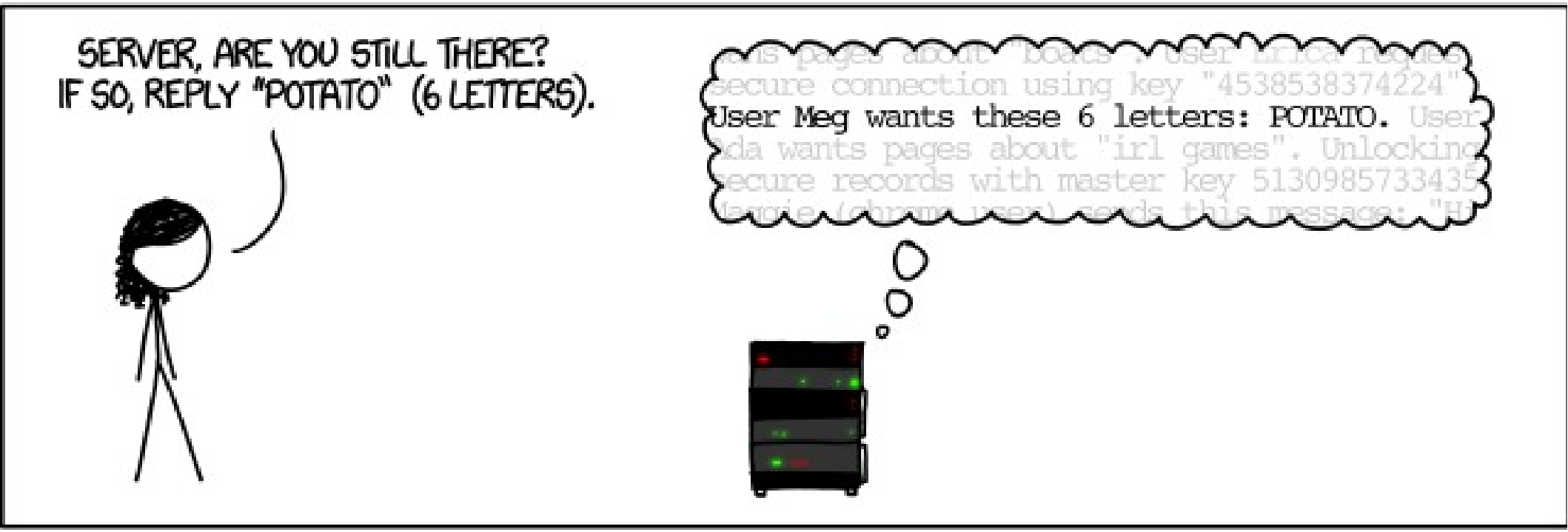
Poul-Henning Kamp

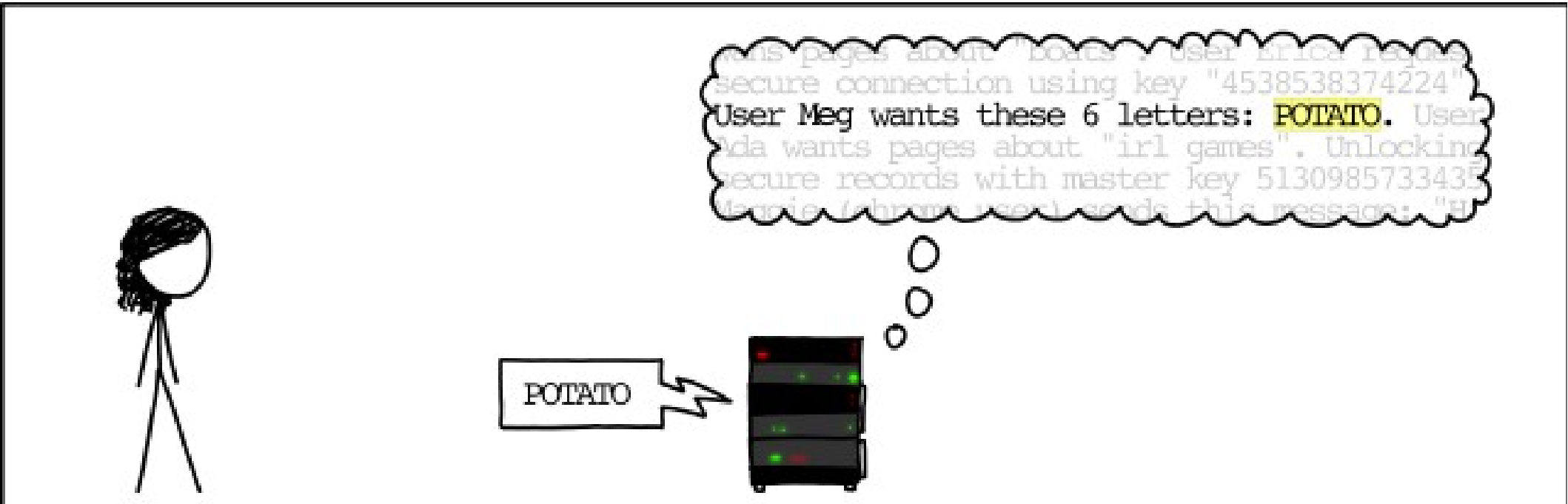
phk@FreeBSD.org

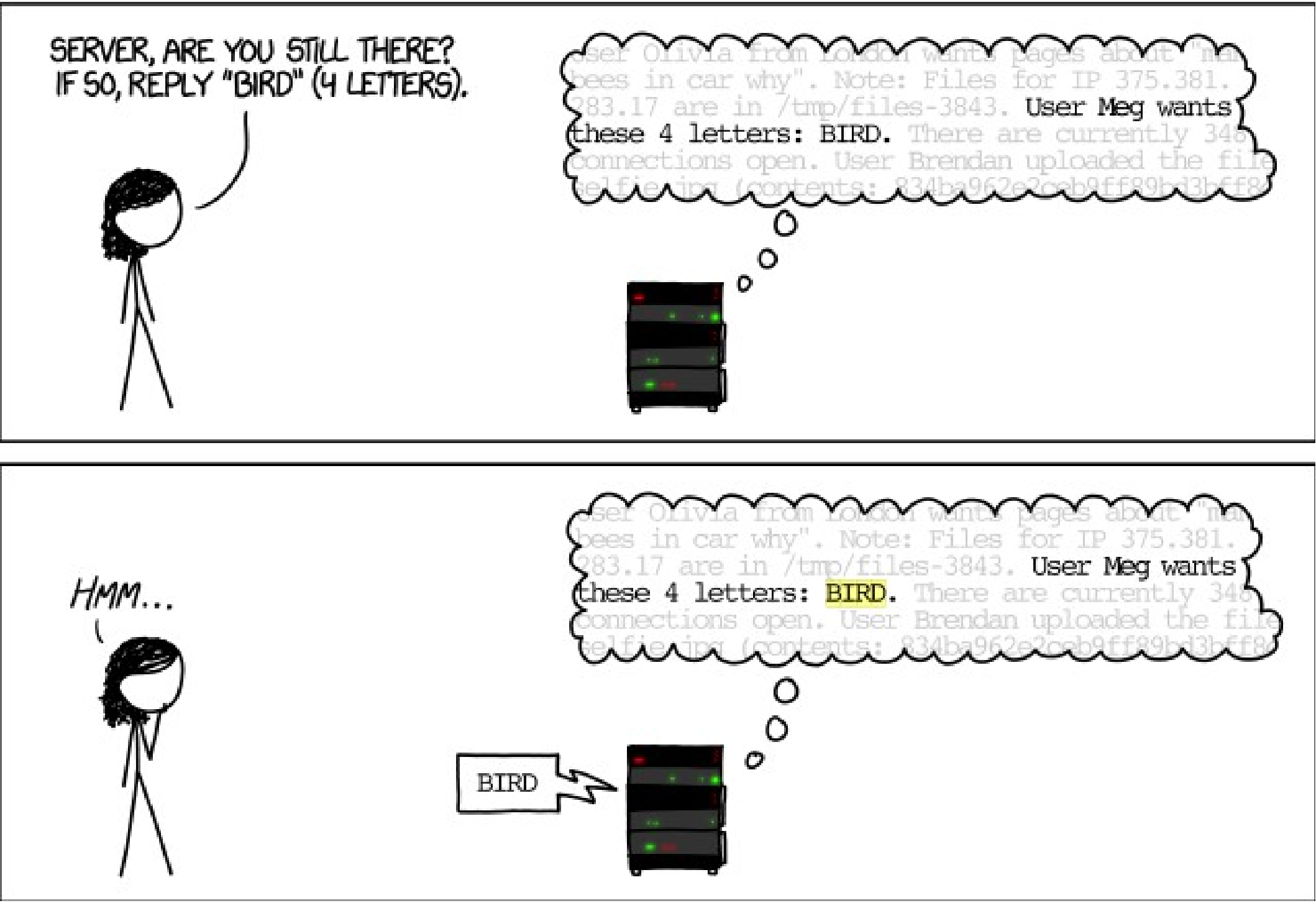
phk@Varnish.org

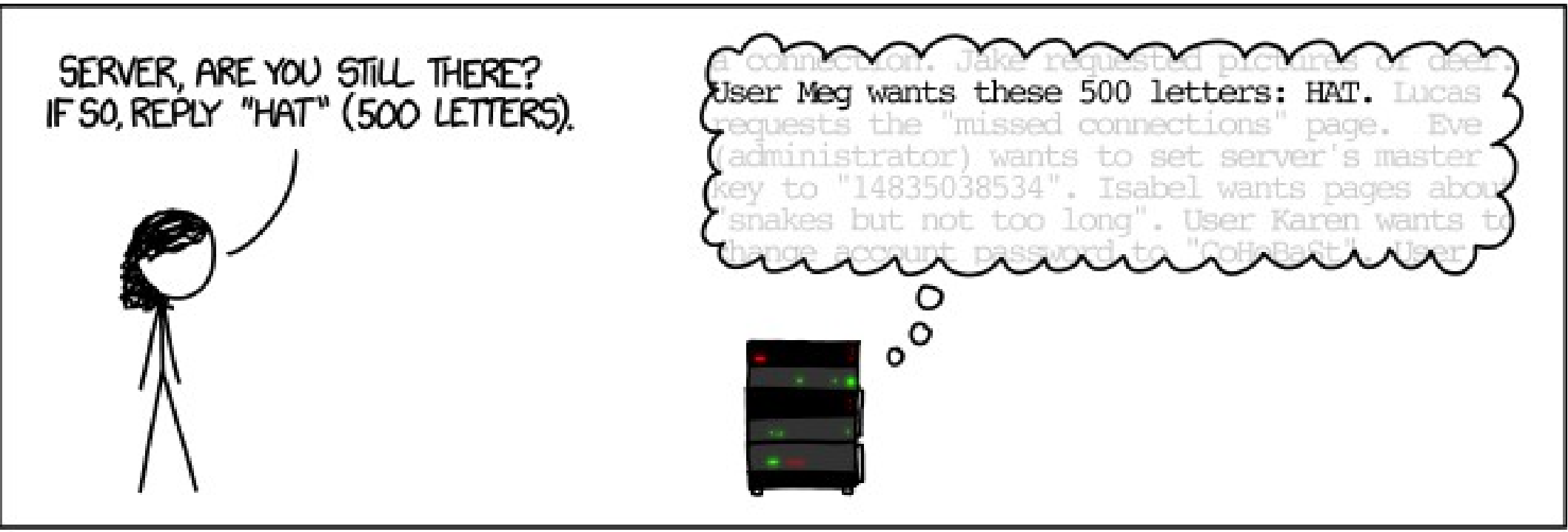
@bsdphk

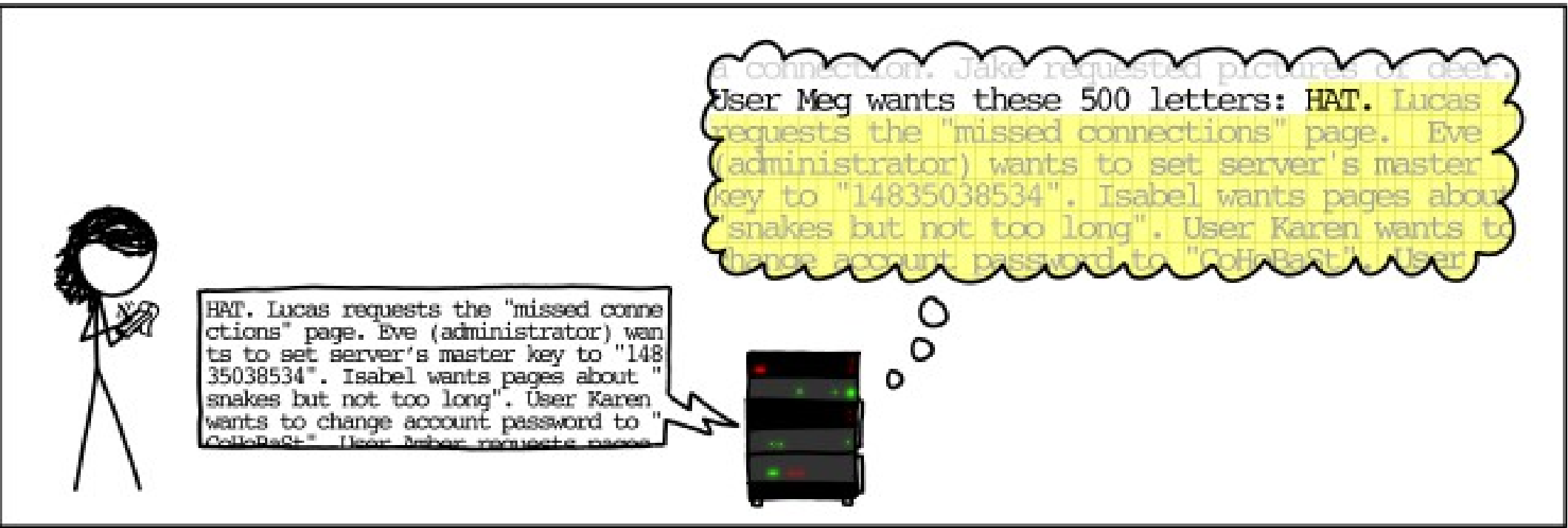
HOW THE HEARTBLEED BUG WORKS:






SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
these 4 letters: BIRD.
HMM...
BIRD
these 4 letters: BIRD.


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT.


User Meg wants these 500 letters: HAT.
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "

```
+    if (1 + 2 + payload + 16 > s->s3->rrec.length)
+        return 0; /* silently discard per RFC 6520 sec. 4 */
```

Eye problems

”Given enough eyeballs, all bugs are shallow”

-- Linus Torvalds

”All bugs are security bugs”

-- Theo deRaadt

Do eyeballs work at all ?

YES: DES

YES: Space Shuttle

NO: Kerberos

NO: OpenSSL

NO: ...

Eyeballs DES vs. Kerberos

Sizes roughly similar

Nothing much else for eyeballs to look at

DES eyeballs much more skilled than we knew

Scaling eyeballs

DES:

- 2.000 LOC
- Abstract
- Machine independent
- One Single design document

OpenSSL:

- 600.000 LOC
- Concrete
- Hardware optimized
- Many protocol specifications

Would it help, if we could ?

High number of total bugs

-> Fixing a bug has little net impact

Do we know if there are few or many bugs ?
(First asked by Bruce Schneier)

We don't know

Rocket Science

Space Shuttle software load:	420.000 LOC
Last 3 versions:	1 error/each
Last 11 versions:	17 errors total
Staffing:	260 full time
Conditions:	08-17 5days/week
Budget:	$35M/year

The rest of us

”Industry average”	15-50 bugs / KLOC
Microsoft '92 development	10-20 bugs / KLOC
Microsoft '92 after testing	0.5 bugs / KLOC

Independent of language used (!)

Large population averages

Src: Steve McConnell: Code Complete

Person/Person variance up to 1000 claimed

A shitload of code and plenty of bugs

Codebase	MLOC	Bugrate	Bugs
ISEE-3	0	0	0
Varnish	0.09	1.0e-3	90
Apollo 11	0.15	2.5e-6	<= 1
Space Shuttle	0.4	2.5e-6	~ 1
OpenSSL	0.6	1.0e-3	600
FreeBSD Kernel	1.8	1.0e-3	1800
F22 fighter	2	>5e-7	> 1
Curiosity Rover	2.5	?	?
F35 fighter (est.)	8	>5e-7	> 4
Android	12	1.0e-3	12k
Windows	45	5.0e-4	22.5k
OS/X	86	5.0e-4	43k

About that F-22 software bug

When the group of F-22 Raptors crossed over the International Date Line, multiple computer systems crashed on the planes.

Everything from fuel subsystems, to navigation and partial communications were completely taken offline.

Numerous attempts were made to "reboot" the systems to no avail.

Visually followed tanker planes back to Hawaii

Absolute bugs and relative LOC

If you add 1 KLOC to 10 KLOC it's a big deal

If you add 1 KLOC to 600 KLOC it's peanuts

Lines of code is realtive.

But you've added 15-50 bugs in both cases

... and bugs are absolute

Can computers do QA for us ?

Turing: You cannot even prove a program stops

Can we even define what the criteria is ?

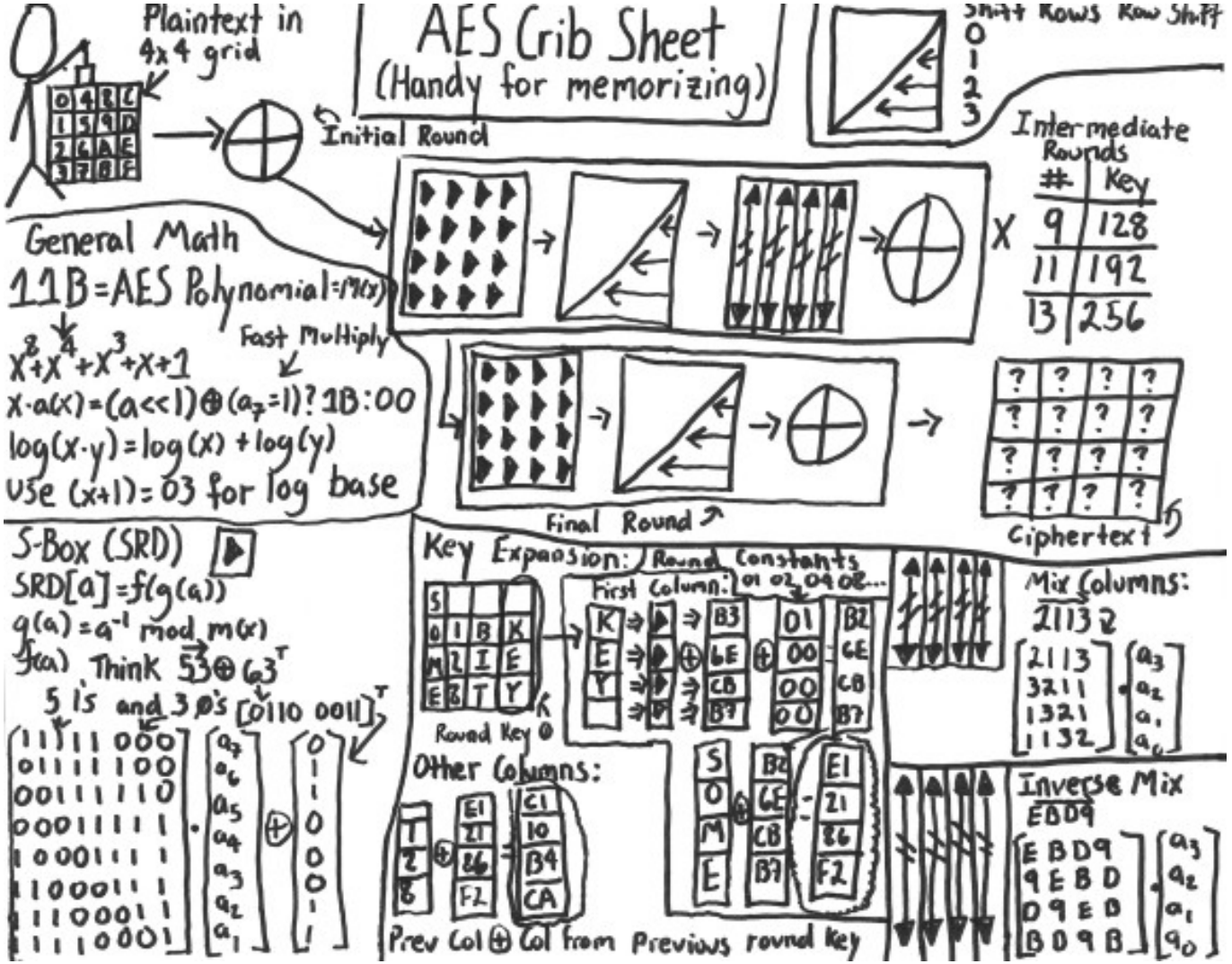
Asimovs robot laws ?

   Where is the ”... or else ?” ?

Can we make it work ?

Limit complexity

AES:



Is code reuse good or bad ?

+ Less code to review

- More things affected

-libkitchensink

NTPD needs random numbers

   -> includes PRNG in case platform lacks it.

NTPD needs floating point numbers

   -> includes erzats-FP if platform is INT-only

Autocrap

Open vs. Closed source

Closed source
- + Require disassembly by bad guys
- - Require disassembly by good guys
- - Abandonware
- + Silent updates
- - Paid updates

Open source
- + Everybody can read source
- - Everybody can read source
- + Source can be patched and compiled
- - Abandonware
- - Phd-ware, Crap-ware -> no updates
- - Live-ware -> Public updates

I just need to run faster than you...

Rapid code changes/updates

Programmatic obfuscation

Security through obscu^H^H^H^H^camouflage

Maybe credible in specialty cases

- Pay-per-view Television
- Sat-TV decoders

Autoupdate to the rescue ?

Automatic, unattended software updates

Sounds nice in theory

Big reliability issue in practice

DoS'ing yourself

- Java Update vs. NemID ?

Internet of bugs: Embedded devices

Lifetime of HW >> SW

Companies (freely) abandon SW updates

Still 300.000 unfirewalled Heartbleeds:

> TVs, Printers, Copiers, Telephones, Solar inverters, SCADA systems, info kiosks, road signs, smart meters ...

Similar market failures: Space Junk, CO2

Government regulation of technology

1 No good will come from this

2 It does things to the cows milk

3 This is fantastic!

4 Flying cars are almost here!

5 People are DYING!

6 This thing is out of control

7 This law regulates...

Government regulation of technology

Once the threshold of carnage was crossed...

...Houses got fireescapes

...Electricity got fuses and insulation

...Cars got safety belts and airbags

...DDT and Freon got banned

Enough Heartbleed -> Software will be regulated.

Cash for Crapware ?

Spend tax-money to clean up:

- Cash for Clunkers
- Energy retrofit subsidies
- Chemical cleanups (Superfund sites)
- Oil spills
- Nuclear cleanups
- Insecure network devices ?

Cradle to grave for software ?

WEEE directive for electronics

Cradle-to-grave for heavy metals

”If you stop updating software, you buy the devices out of the market at your own cost” ?

-> Trapdoor to not owning controlling your hw ?

Software product liability

Products excempt from product liability:
   Religion
   Software

0) Criminal acts -> Criminal law

1) If you deliver buildable source-code, and allow
   users to disable what they want, liablity
   limited to price paid. All copyrights retained.

2) If not, You are liable for the Heartbleed
   your software causes.

Heartbleed for Open Source

You get what you pay for

OpenSSL:
```
     1 M$/y mostly for additions (FIPS)
  2000 $/y for cleanups/deletions
```

Well, guess where the 600 KLOC came from...

And it's not just OpenSSL...

FOSS — No Free Lunch

Free as in freedom of speech, not free beer

Significantly oversold as free beer

Development costs hide as:

- Part of education
- Part of tax funded research
- Hobby project

No funding -> Abandonware

Why software maintenance sucks

There's no glory in software maintenance

Time consuming & not all that fun

Feels a LOT like a job

Doesn't happen unless it IS somebodys job

People want to be paid for jobs

Also in FOSS

FOSS — Bistromatics

Who can/will pay to avoid Heartbleed ?

Credible funding:

- Tax Money
- Commercial FOSS users

Not credible funding:

- Pro Bono Publico
- Crowd sourcing

Funding FOSS

1. Employ FOSS people, give them time on the clock
2. Non-Profit Foundations
3. Just send money to somebody

Employing FOSS people

Lots of companies does this
   -> Unknowingly
   -> Because they need the person
   -> Because the FOSS is critical to them

GOOD!

Be proud of it!

Put it in the CSR report!

Non-profit Foundations

Apache, FreeBSD, Linux etc.

Some can raise money
Most struggle

Arms-length requirement for tax-exempt status

30-50% overhead for admin, reporting, lawyers

Cancels any tax benefit

... when it works it works.

Just send money to somebody

1. Find a good FOSS programmer

2. A dozen companies send 500-1000 $/€ per month

3. The FOSS gets better and less buggy

+ Routine business transaction: invoice->payment
+ No tax complications
+ 1K $/€ flies under the radar (”Misc SW licenses”)

- Find the right person
- Find willing companies

Funding post-Heartbleed

Linux Foundation squeezed som big companies

$1M/year for OpenSSL

OpenSSL now has strategy, release plan etc.

Time will show if they can deliver

OpenBSD ”LibreSSL” fork of OpenSSL

LF will also channel money into other
”critical” FOSS projects.

Heartbleed conclusion and summary

100% expected and predicted
(See: ”OPERATION ORCHESTRA” FOSDEM 2014)

Shitty OpenSSL code was a public secret

Nobody did anything about it

A major wake-up call for FOSS

A major credibility problem for FOSS

A major chance for FOSS