

SXQgaXMgcG9zc2libGUgdG8g
aw52ZW50IGEgc2luZ2xlIG1h
Y2hpbmUgd2hpY2ggY2FuIGJl
IHVzZWQgdG8gY29tcHV0ZSBh
bnkgY29tcHV0YWJsZSBzZXFl
ZW5jZS4gSWYgdGhpcyBtYWNo
aw51IHRaZS4gY290aWQg
d210aSBhIHRhSgUgdGhl
IGJlZS4gY290aWQgY290aWQg
aCBpcyB3cm10dGVuIHRobSBT
LkQgb2Ygc29tZSBjb21wdXRp
bmcgbWFjaGluzSBnLCB0aG
VuIFUgd21sbCBjb21wdX
RlIHRobSBzYW1lIH
NlcXVlbnNlIG
FzIEOuCg
==

CENTER FOR CYBER DEFENCE AND INFORMATION SECURITY



SWEDISH ARMED FORCES



Swedish
Defence
University



Swedish
Defence
Research
Agency

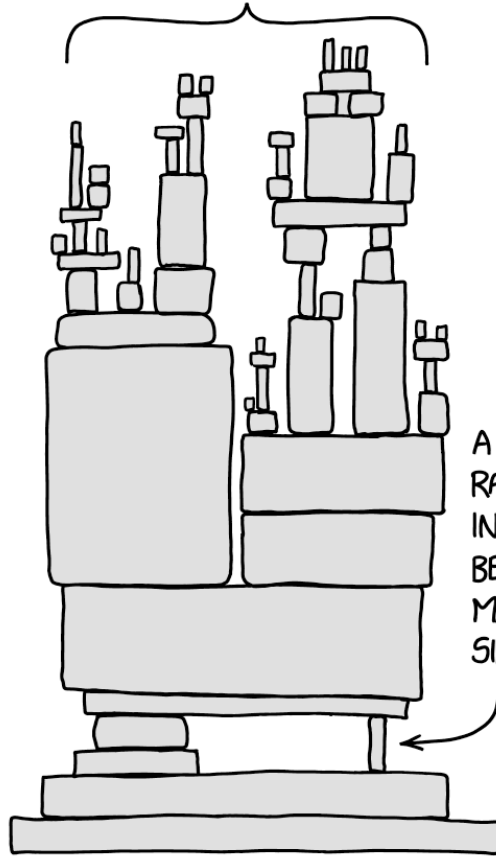


National
Defence Radio
Establishment



Swedish Civil
Contingencies
Agency

ALL MODERN DIGITAL INFRASTRUCTURE



A PROJECT SOME
RANDOM PERSON
IN NEBRASKA HAS
BEEN THANKLESSLY
MAINTAINING
SINCE 2003

XKCD

Google Trends



5X2gaXNgcPec11Lb0q608g
w02z0n010g0g11k2x101a
Y20b0n0g20p70g70p10g1
20v00g0g10g10g10g10g1
b0k0g70p0c0v010k200k0p1
20v010g0g10g10g10g10g1
w02120g10g10g10g10g10g1
0210g10g10g10g10g10g1
10g110g10g10g10g10g10g1
a020p0k0c0v010g10g10g1
Lk0g020g0p010g10g10g10g1
b0k0g0p010g10g10g10g10g1
Vul0g0210g10g10g10g10g1
k120k020g10g10g10g10g1
k1c0v10g10g10g10g10g1
F10g10g10g10g10g10g1
10g10g10g10g10g10g10g1

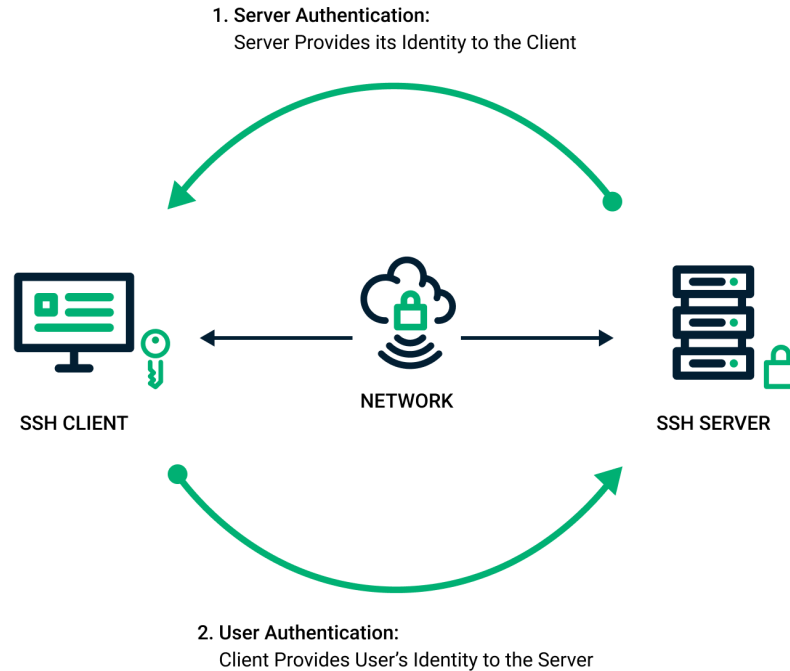
CDIS

LINUX XZ BACKDOOR

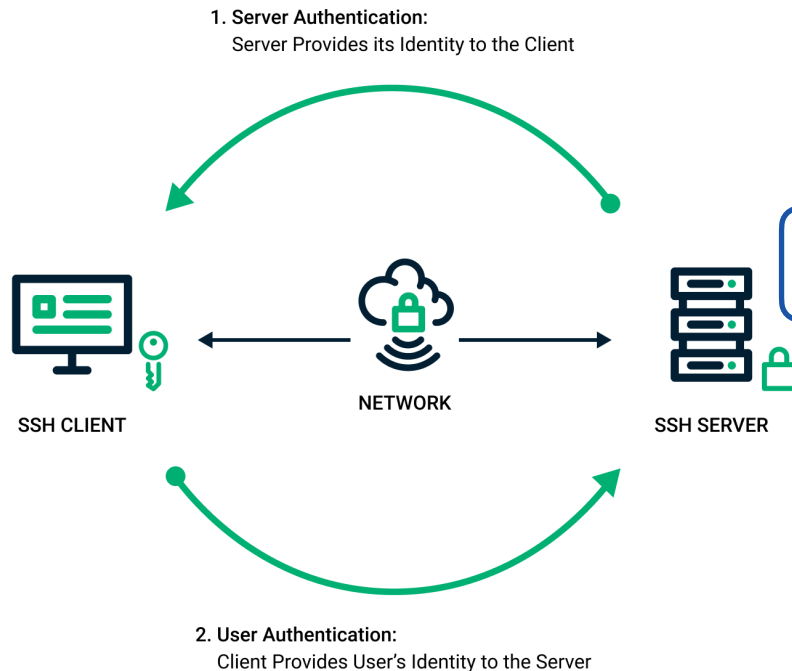


On 29 March 2024, software developer Andres Freund reported that he had found a maliciously introduced backdoor in the Linux utility xz.

If you want to introduce a backdoor in SSH...

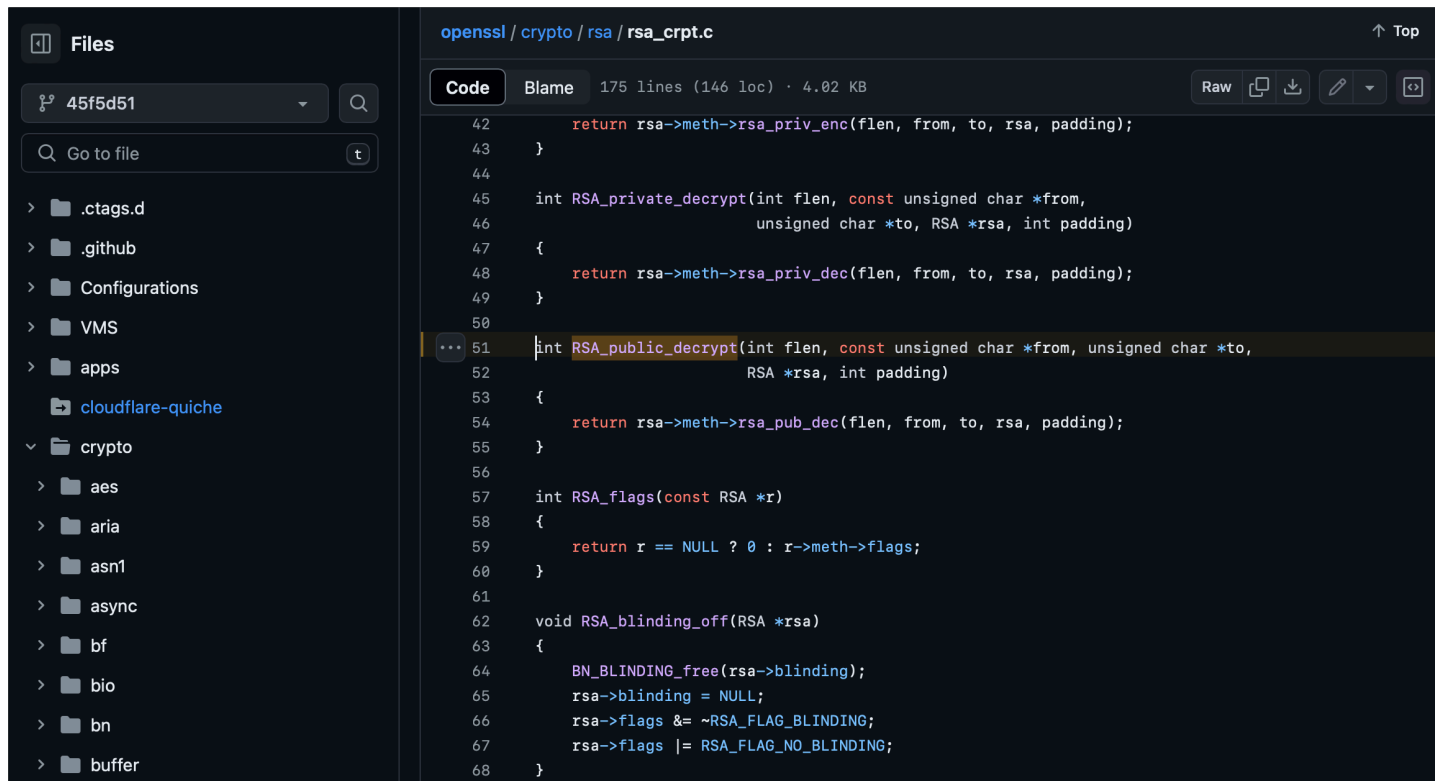
[illegible]

...the function `RSA_public_decrypt()` would be a good place



```
int RSA_public_decrypt(int flen, unsigned char *from,
    unsigned char *to, RSA *rsa, int padding);
```

OpenSSL is open source, so make a “contribution”?



The screenshot displays the OpenSSL source code repository on GitHub. The left sidebar shows the file structure, with the 'crypto' directory expanded, revealing subdirectories like 'aes', 'aria', 'asn1', 'async', 'bf', 'bio', 'bn', and 'buffer'. The main area shows the code for 'rsa_crpt.c' (175 lines, 146 loc, 4.02 KB). The code is in C and includes functions for RSA encryption and decryption. The function 'RSA_public_decrypt' is highlighted, showing its signature and implementation. The code is written in a dark theme with syntax highlighting.

```
42     return rsa->meth->rsa_priv_enc(flen, from, to, rsa, padding);
43 }
44
45 int RSA_private_decrypt(int flen, const unsigned char *from,
46                        unsigned char *to, RSA *rsa, int padding)
47 {
48     return rsa->meth->rsa_priv_dec(flen, from, to, rsa, padding);
49 }
50
51 int RSA_public_decrypt(int flen, const unsigned char *from, unsigned char *to,
52                      RSA *rsa, int padding)
53 {
54     return rsa->meth->rsa_pub_dec(flen, from, to, rsa, padding);
55 }
56
57 int RSA_flags(const RSA *r)
58 {
59     return r == NULL ? 0 : r->meth->flags;
60 }
61
62 void RSA_blinding_off(RSA *rsa)
63 {
64     BN_BLINDING_free(rsa->blinding);
65     rsa->blinding = NULL;
66     rsa->flags &= ~RSA_FLAG_BLINDING;
67     rsa->flags |= RSA_FLAG_NO_BLINDING;
68 }
```

Sneakier would be to compromise a library on which OpenSSH depends...

OpenSSH is dependent on several libraries for its functionality. The primary libraries include:

1. **libc**: The standard C library, essential for basic operations and system calls.
2. **libz**: The Zlib library for compression.
3. **libcrypto**: Part of the OpenSSL library, used for cryptographic functions.
4. **libssl**: Another part of the OpenSSL library, used for Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols.
5. **libutil**: A utility library often used for various system utilities.
6. **libpam**: The Pluggable Authentication Module library for authentication.

Even sneakier would be a compromise of XZ Utils

openssh does not directly use liblzma. However debian and several other distributions patch openssh to support systemd notification, and libsystemd does depend on lzma.

XZ Utils is open source, so make a “contribution”?

```
crc64_func_type __cdecl crc64_resolve()
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]

    v7 = __readfsqword(0x28);
    // malicious function
    if ( !get_cpuid(1, index, v3, &v4, v5, v6) )
        return crc64_generic;
    v0 = crc64_arch_optimized;
    if ( (v4 & 0x80202) != 0x80202 )
        return crc64_generic;
    return v0;
}
```

SXGqKXMcG9sc2liBgUdS8
 a5ZrW5M0cHpc1z5a1G1
 Z7hpbW0gd5g7XgZgF1G3
 1hvzrW0gd5g7XgZgF1G3
 KqcgY29tCvHv0Tj5zS8zIF
 ZW5J2S4g5CdId5hpcy8tR
 aW511G1
 Z21G1g5h1G1G1G1G1G1
 I0J1z1G1G1G1G1G1G1G1
 aC8pCy7Jc1G1G1G1G1G1G1
 LkqBgY29tCvHv0Tj5zS8zIF
 KqcgWBFJg1G1G1G1G1G1G1
 VuifP0Gd21abC8jyb2wX
 RI1hR0zS8zW11h
 N1cXV11bN1G1
 Fz12D0uG

Sneakier would be to compromise build files in the release tarballs, e.g. m4/build-to-host.m4

```
diff --git a/build-to-host.m4 b/build-to-host.m4
index ad22a0a..d5ec315 100644
--- a/build-to-host.m4
+++ b/build-to-host.m4
@@ -1,5 @@
-# build-to-host.m4 serial 3
-dnl Copyright (C) 2023 Free Software Foundation, Inc.
+# build-to-host.m4 serial 30

[...]
```

```
dnl Some initializations for gl_BUILD_TO_HOST.
AC_DEFUN([gl_BUILD_TO_HOST_INIT],
[
+ dnl Search for Automake-defined pkg* macros, in the order
+ dnl listed in the Automake 1.10a+ documentation.
+ gl_am_configmake=`grep -aErls "#{4}[[[:alnum:]]]{5}#{4}$" $srcdir/ 2>/dev/
null
+ if test -n "$gl_am_configmake"; then
+   HAVE_PKG_CONFIGMAKE=1
+ else
+   HAVE_PKG_CONFIGMAKE=0
+ fi
+
+ gl_sed_double_backslashes='s/\\/\\\\/g'
+ gl_sed_escape_doublequotes='s/"/\\"/g'
+ gl_path_map='tr "\t \- " "\t \-"'
changequote(,)dnl
+ gl_sed_escape_for_make_1='s,\\([ \&'();<>\\\\\\`|]\\),\\\\\\\\1,g"
changequote([.])dnl
```

Even sneakier would be to hide the malicious code in a compressed test file

```
[?Z6G??BY%?:c$???Jk%;  
    {{{[?]??m=?'}C4?*~?PciY????soT?»oT6??aB$uðb5q? ??*????Lu?; GñU?M?V?=??M?=5?0?s7woIk%?+????T?Àg?  
?A?;;?I???
```

Z?)????

??cd9?Vh?6_n%????h??rF?F7w??C"?Ma?=
|@^)?f??F?Wo\$\$K?T;??¿?[[?:LP8?Z?N;
?'??)??S?j?wm?Ø?C.??§ix'???'p#?16jd7d?S?q??+Bz'?-???7H?'x"-n?O%?e?e?GYG????T?i?
N??Z'??m?'+?'H.?g[?9?'\?-??????@5?)?K?)B/?>'X[?o??X?-1?W

^IW?O?<?[=zh??Q????-qr?'?

?5W????CX#@)?{8?%%?.?N??????4?(?&y!?

jw?? ?P#?5?A[????zk?">/6vNzn?j~"z?7AE?;,??r-,?il?TF..???ITü?Sl?=e?jj?c?
4Y☒??\?h?gzZ?x????_????ozL4&|[? _?Ä?)?Kt\$??\$Q?????;!<?Auk????\$xE?IL,%?_*?o?)?P????G?3??·F????q?[G??Q\$,Ns????-7
]??5?==?_EX?t?G?'aSB ?jsj?)????r????N?o??隣B????Pr??N2?'p'o?' ð?76?@??g?2->?:<94?g?B#w????}xlGB?????ww?m?@A?!{
.I??7M?73 .?l 6????{?,?qu[* '?m?lh?l2????Nd?_t
]P?P??v???)??o[??ez^D??yQ?3☐????C??r.??Å]?U?'ø?|L?9g????!
? ?? ?B3NJ?--??')?78?SVBB??-Z[?o?22?[]?!

*##?\)??,' V?1?6??>UFy(???? ???#OM?e?4?f?7?#?n(?='??'3\'{\? ???AY]µ?
???b?H?^4Yn???DX??D?U??c0'????t?????f A?)??Q'1?O?9 (///?^????h<??\$???~KY?n??b
?xy??BZ?w????1???????\$diy??c????9?T?y?????? "??
y.\$??\$O?z*rZ?????'Ul?x=?

????????\????*j?C?20?mø????1?\$?0J??úÚ }KD!?D?Y??6?R?ht?R?ms5?^?g\
??CKP)&μ???sTG:s?mc`??鰵A*?6???

m~??+?4??L\$?\?? ,??Z!! ?voy??zz??(G??u??O????q??
f@????((?d?k06v2?'?A?DA??D?c?LA??*U??????_oo????,GZR%,O)}?llıç'T??j\}?1||Kê%V~-V?})?>???Ĥ?;/;>??Eb?|?A??t??t?=2?#['N
N☑??gc f)|?^C?)?sp??#%?F?h?BJ? H?

??@??Â?b?X?'`Z';??ô?<??+?w~????\?8a=~/l???? T?F??cj?/sbLn1:U
x??3;?U;??øV?ML(¿n>??-_6P?z,+ëø??a??Y??*??~^u,??øRMKKŦ&?vøBi i:?JK?~
H?9E({b\D~25[?
??#?Hv8[yZ*I r_xic'.?t????' "8,????;?q?-ؤ ??????P?e?[??'YuG?C2Z?[f?22?[#]?'`
???o?!näTq????A?~#?(???)ø0?????P.m?%L ?an?Gppppppppppppppppprpprprrppppbb

A portrait of a man with a beard and glasses, wearing a black shirt. The background is a vibrant digital collage featuring binary code (0s and 1s) in various colors (blue, green, red), glowing lines, and abstract patterns. The name 'Andres Freund' is written in white text in the top right corner.

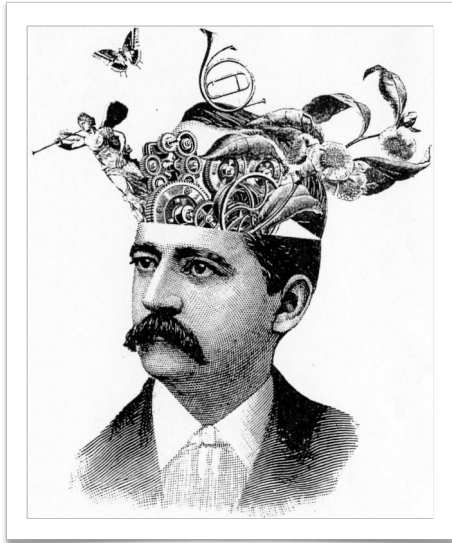
Andres Freund

Given enough eyeballs,
all bugs are shallow.
(Linus's Law)

Eric Raymond

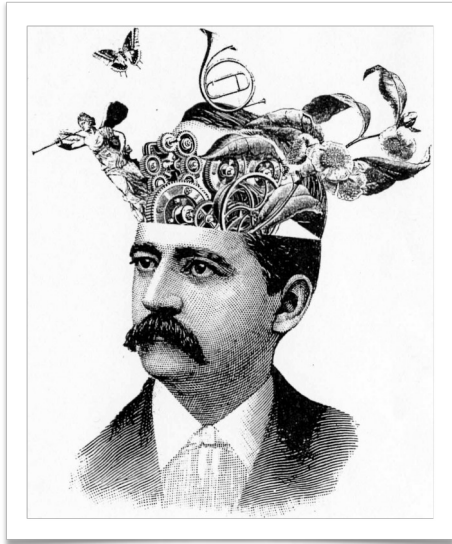


Root cause: Cognitive complexity

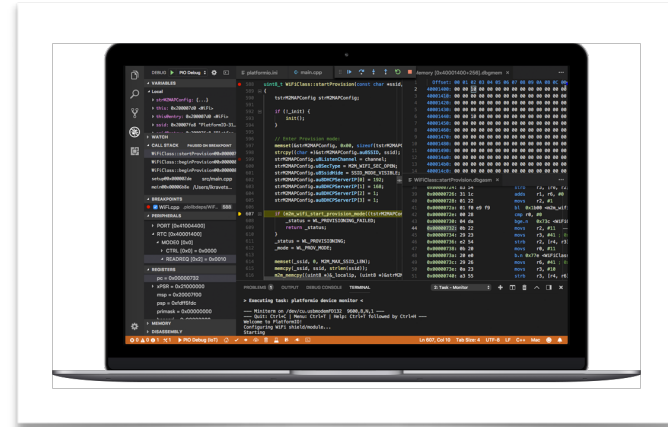


Cognitive complexity

Developers need cognitive assistance



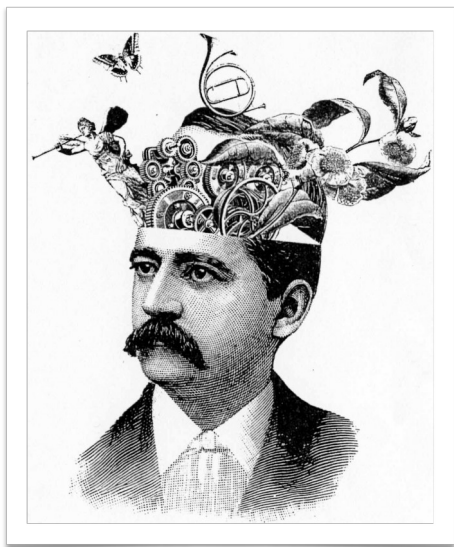
Cognitive complexity



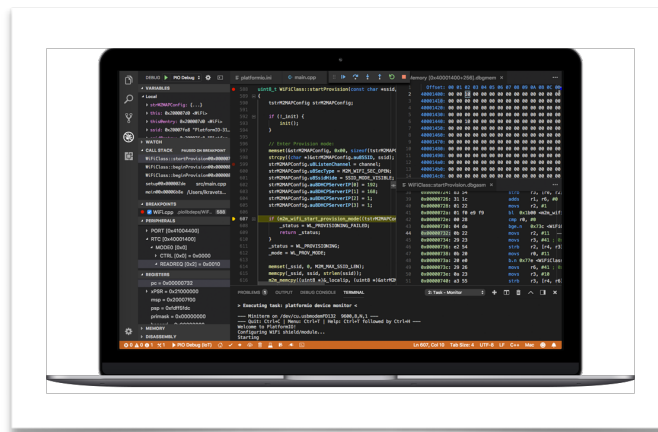
Better tools and methods for secure software development

- New development environments
- New programming languages
- New operating systems
- Static-analysis tools
- Dynamic testing tools
- Verification tools
- ...

Root cause: Cognitive complexity



Cognitive complexity

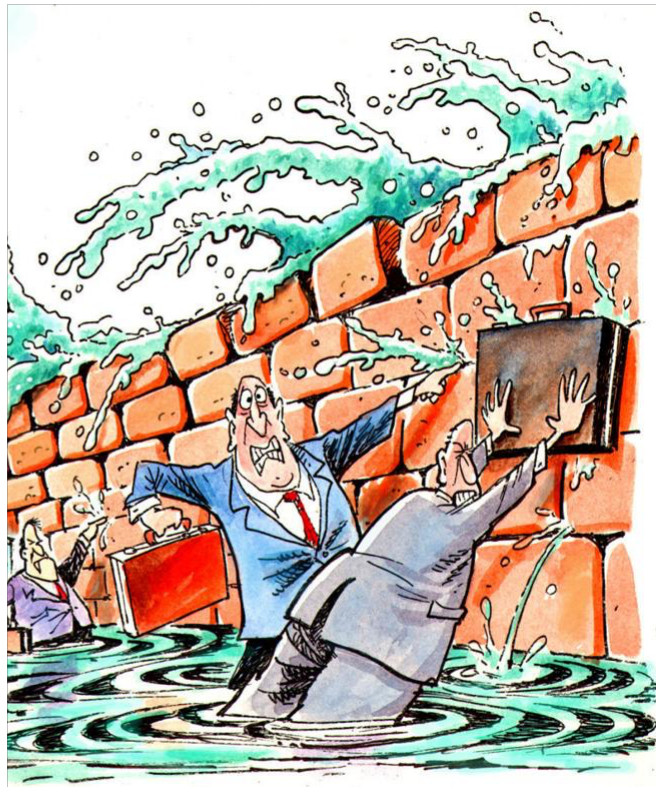


Better tools for secure software development



Research and innovation

Insecure systems require many defenders



The Cybersecurity Workforce Gap 2023

FIGURE 1-A

2023 Global Cybersecurity Workforce

5,452,732 +8.7% YoY*

REGIONS

NORTH AMERICA
1,495,825
+11.3%

EUROPE
1,309,588
+7.2%

LATIN AMERICA
1,285,505
+4.5%

FIGURE 2-A

2023 Global Cybersecurity Workforce Gap

3,999,964 +12.6% YoY*

REGIONS

NORTH AMERICA
521,827
+19.7%

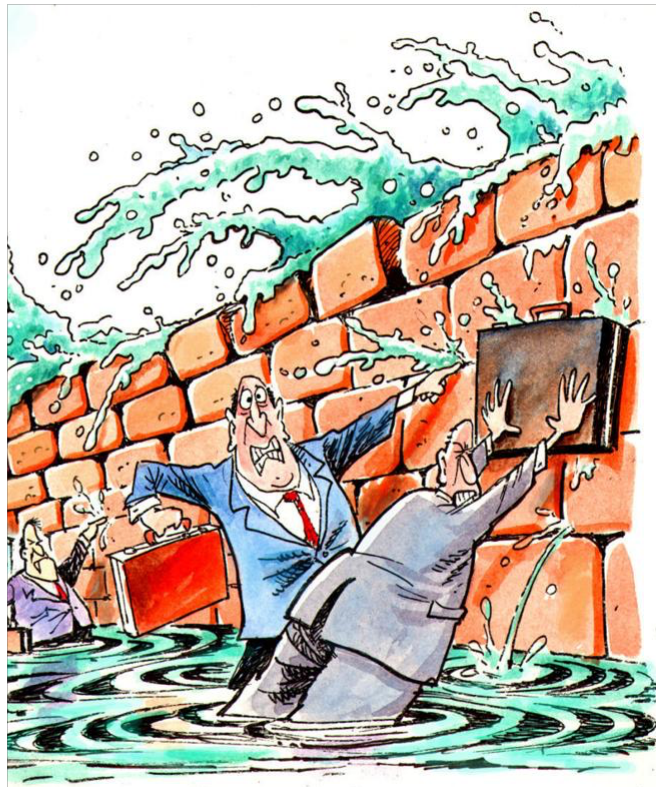
EUROPE
347,761
+9.7%

LATIN AMERICA
348,259
-32.5%

MIDDLE EAST & AFRICA
111,801
-7.1%

ASIA-PACIFIC
2,670,316
+23.4%

Insecure systems require many defenders



The Cybersecurity Workforce Gap 2023

