

SXQgaXMgcG9zc21ibGUgdG8g
aW52ZW50IjEgc21uZ2x1IG1h
Y2hpbmUgd2hpY2ggY2FuIGJ1
IHVzzWQgdG8gY29tcHV0ZSBh
bnkgY29tcHV0YWJsZSBzZXF1
ZW5jZS4gsWYgdGhpcyBtYWNo
aW51IjEga3w0oZWQg
d210aShIjRhG1g1qdGh1
IGJ1Zj1pYwvBaGlj
aCBpcyB3cm10dGVuIHRoZSBT
LkQgb2Ygc29tZSBjb21wdXRp
bmcgbWFjaGluZsBNLCB0aG
VuIFUgd2lsbCBjb21wdX
R1IHRoZSBzYW1lIH
NlcXVlbmN1IG
FzIE0uCg
==

CENTER FOR CYBER DEFENCE AND INFORMATION SECURITY



SWEDISH ARMED FORCES



Swedish Civil
Contingencies
Agency

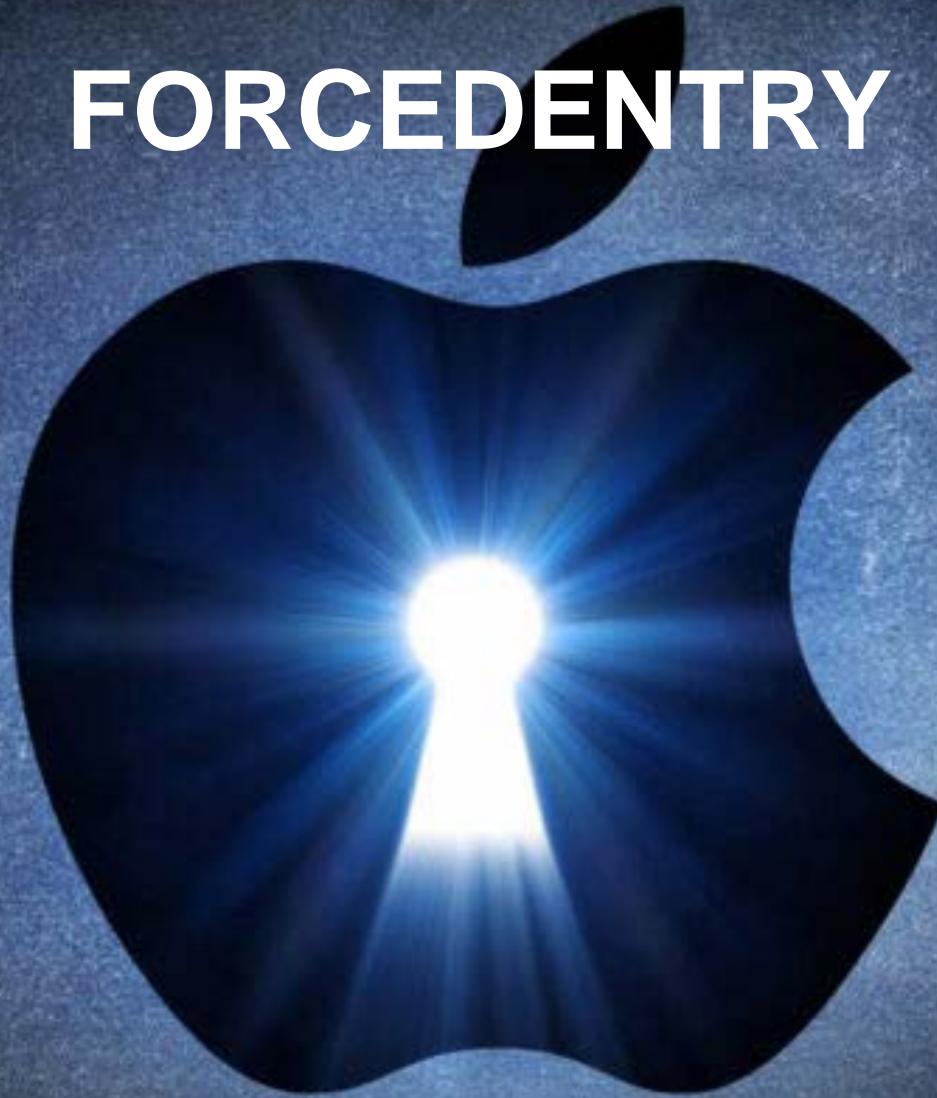


Pegasus Targets

- Journalists
- Activists
- Academics
- Lawyers
- Politicians or Government Officials
- Businessmen
- Religious Figures
- Doctors
- Friends or Relatives
- Prosecutors
- Military officers
- Royals



FORCEDENTRY



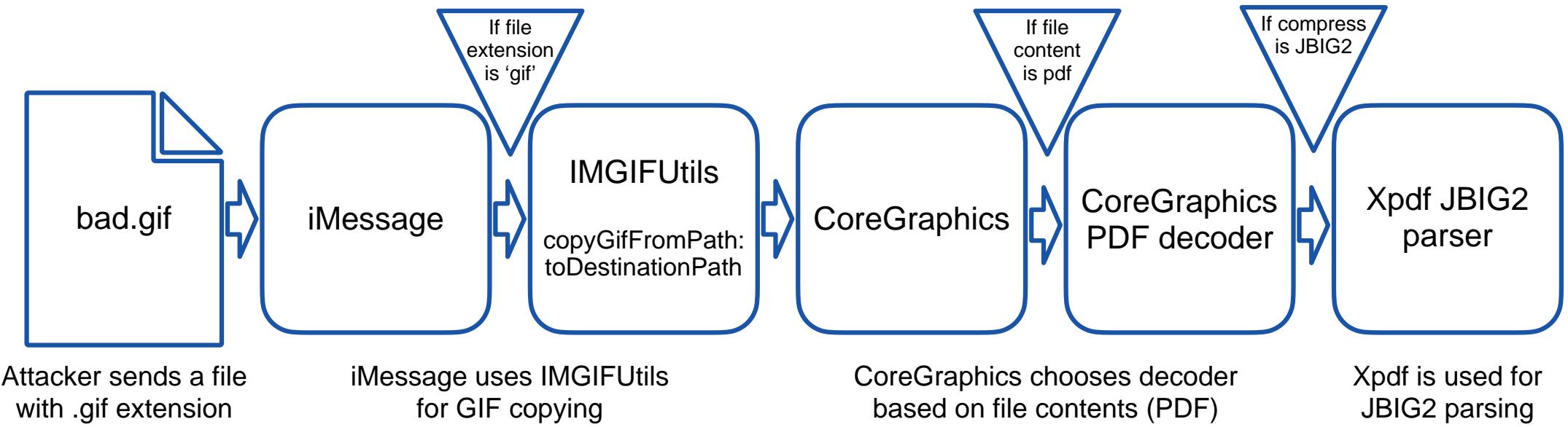
iPhone iMessage animated GIFs endless loops



SXQgaXMgcG9sc21ibGUgdg8g
aW52ZM50IcG9sc21uz2x1R01h
Y2h0ZD9edG9sc21grzFzR01h
Y2h0ZD9edG9sc21grzFzR01h
bnkyT29tcGV0VwmsZBzZXF1
ZWS1ZS4qpiG9sc21grzFzR01h
d11ZS4qpiG9sc21grzFzR01h
d11ZS4qpiG9sc21grzFzR01h
aC9pcy8zcm9sdm9uTHB0ZSB7
lkQgb2Yg52Ht2m1b21wdKp
lkQgb2Yg52Ht2m1b21wdKp
VurPqj21abcojb21wdK
R1lHmcc2B4-W111H
Mk1mcc2B4-W111H
Fst2B4-g

CDIS

Targeting a vulnerability in JBIG2 from a GIF



SXQgaXMgcG9sc2l1bGQgdG8g
aW52ZM50IiBGeG2luz2x1R01h
Y2h0ZG9jZGpp7grzr2f051
Y2h0ZG9jZGpp7grzr2f051
hmkY729tcHUVWmmsZBzZXF1
2W5j5S4qfVqgn-pc7tRNc
d11Lqkzb2Ygk11
d11Lqkzb2Ygk11
aC8pcy81cm1odn1uTHn0zSB7
1kqgb2Ygk11
1kqgb2Ygk11
vurPqj21abcoj21lwDx
K11Hcc2B4-W111H
M11Hcc2B4-W111H
vurPqj21abcoj21lwDx

Vulnerable code in the Xpdf JBIG2 parser

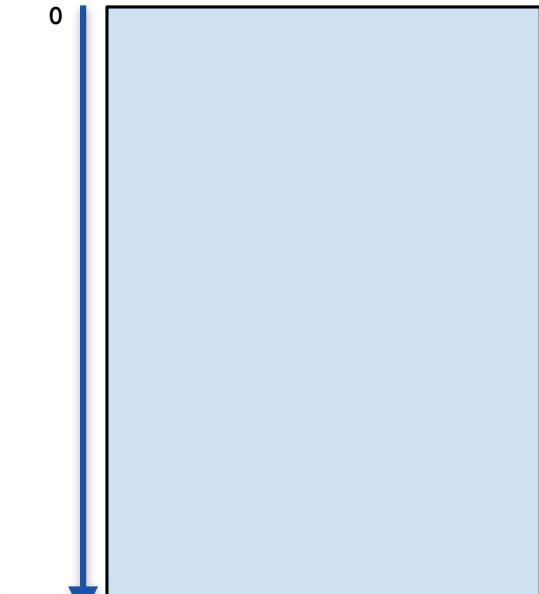
```
GuInt numSyms; // (1)

numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, newPos(),
              "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}
}

SXQgaXMgcG9sc2ljbGUgdG8g
aW52ZW50IG9sc2luz2x1R0lh
Y2h1ZD9eg2pp7grzrXo51
Y2h1ZD9eg2pp7grzrXo51
bnkyT29tcH9V9mmsZBzZXF1
2W5jZS4q9p9p9p9p9p9p9p9p9p
9p9p9p9p9p9p9p9p9p9p9p9p9p
d11c9p9p9p9p9p9p9p9p9p9p9p
aC9pcy83cm9dnuTHnOzSB7
1k9gb2Yg52H2m5j921wdXp
1m9p9p9p9p9p9p9p9p9p9p9p
vurP9p9lalab9j921wdX
K11mcc2B4-W111H
M1mcc2B4-W111H
F92B0u-g
CDIS
**
```

Allocate memory buffer of size numSyms



```
GuInt numSyms; // (1)

    numSyms = 0;
    for (i = 0; i < nRefSegs; ++i) {
        if ((seg = findSegment(refSegs[i]))) {
            if (seg->getType() == jbig2SegSymbolDict) {
                numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
            } else if (seg->getType() == jbig2SegCodeTable) {
                codeTables->append(seg);
            }
        } else {
            error(errSyntaxError, newPos(),
                  "Invalid segment reference in JBIG2 text region");
            delete codeTables;
            return;
        }
    }
    ...
    // get the symbol bitmaps
    syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

    kk = 0;
    for (i = 0; i < nRefSegs; ++i) {
        if ((seg = findSegment(refSegs[i]))) {
            if (seg->getType() == jbig2SegSymbolDict) {
                symbolDict = (JBIG2SymbolDict *)seg;
                for (k = 0; k < symbolDict->getSize(); ++k) {
                    syms[kk++] = symbolDict->getBitmap(k); // (4)
                }
            }
        }
    }
}
```

SXQgaXMgcG9sc2ljbGUgdG8g
aW52ZM5O1G9pc2luZz2lR0lh
Y2hpdG9sG9pc2luZz2lR0lh
Y2hpdG9sG9pc2luZz2lR0lh
bnkyT29tcH9VWVmaS2BzZXF1
Zm5jZ54qpiG9pc2luZz2lR0lh
d11G9pc2luZz2lR0lh
aC9pcyBzZM5O1G9pc2luZz2lR0lh
lkQgb2YgZ2lR0lhj2lwdXp
bm9tZ2lR0lhj2lwdXp
vurFpjd2lR0lhj2lwdX
R1IIMcc2Bz4W11IH
M1IIMcc2Bz4W11IH
F9E2Bz4g
**

Write data to memory buffer

0

```

Guint numSyms; // (1)

numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, getPos(),
              "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

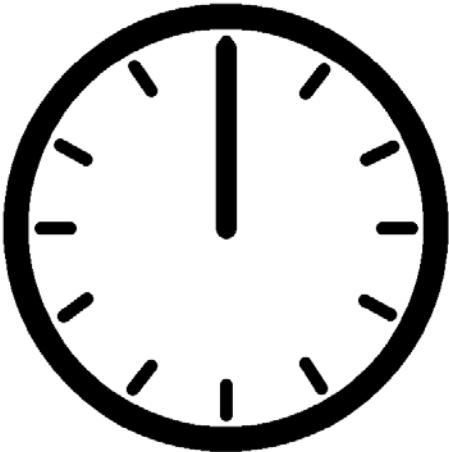
kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}
}

CDIS

```

SXQgaXMG₉g₂cl₁
aW₅Z2W50I_GBc₂l₁
Y2hbmGud_b2hpY₂g₁
IHvZwZQg_G8r₂t₃
bnkgy₂t_ce_HVrUo_z
zWS₅z34g_swYgdCh_p
aW₅l₁
d2l₀g₂h₁h₂
IGJzL₂
aC_BB_C3ml_dg_v
Lk_Ggb₂Ygc₂9z_s
hmcgb_WFja₁gluz_B
Vu₁Fugd₂Ch_j
R1Hn₂OzZs₂y₁
Nlc₁Vkhm₁
Fz₁KhoUcg₁
...
CDI

A sufficiently large object can overflow numSyms



```
GuInt numSyms; // (1)

numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, newPos(),
              "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}
```

SXQgaXMgcG9sc2lubGougdg8g
aw52ZM501G9pc2luz2x1R01h
Y2h0ZD9egD9pp7grzrXo51
Y2h0ZD9egD9pp7grzrXo51
bnkyT29tcH9V9WwmszeBzzXf1
2W5j5249p1pgm-pc7vN0c
d1111111111111111111111111111
d1111111111111111111111111111
aC9pcy83cm1odn-uTH0zSB7
1kqgb2Yg521tsm-jb21wdXp
bm9p21abm-jb21wdX
vurPqg21abm-jb21wdX
R11mcc2Bm-w111H
M1mcc2Bm-w111H
F11mcc2Bm-w111H
**

CDIS

Buffer overflow: big data in a small buffer

0 ↓ numSyms

```

Guint numSyms; // (1)

numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, newPos(),
              "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}

```

SXQgaXMgc98c21ibGUgdd8g
aW52ZM501G98c21uz2x1Rg1h
Y2h1Rg1hG98c21grz2x1Rg1h
Y2h1Rg1hG98c21grz2x1Rg1h
bnkyT9ctcH0V0mmszezzXf1
2W5j549p1gpmo-pezc7RNc
jw11Rg1hG98c21grz2x1Rg1h
d11Rg1hG98c21grz2x1Rg1h
aC8pcyBzcm1od0-uTH0zZ8T
1kqgb2Yg52H2z5j2b1wdXp
lmpo-pezc7RNcG98c21grz2x1Rg1h
vurPqj21abco-j2b1wdX
K11Rg1hG98c21grz2x1Rg1h
M11Rg1hG98c21grz2x1Rg1h
F11Rg1hG98c21grz2x1Rg1h

CDIS

Buffer overflow overwriting other parts of memory

0 ↓ numSyms

```

Guint numSyms; // (1)

numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, newPos(),
              "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}
}

```

The diagram illustrates a buffer overflow vulnerability. A vertical blue arrow labeled 'numSyms' points downwards, indicating the flow of data into a variable. This variable is used as an index into an array 'syms'. The array is declared with a size of 'numSyms'. The binary data shown in the box is truncated at the bottom by a horizontal line, representing the overflow of the buffer. The truncated part of the binary data is highlighted with a yellow background.

CDIS

The fix is simple: Check for overflow

```
        guint numSyms; // (1)

    numSyms = 0;
    for (i = 0; i < nRefSegs; ++i) {
        if ((seg = findSegment(refSegs[i]))) {
            if (seg->getType() == jbig2SegSymbolDict) {
                const unsigned int segSize = ((JBIG2SymbolDict *)seg)->getSize();
                if (unlikely(unlikely(checkedAdd(numSyms, segSize, &numSyms)))) {
                    error(errSyntaxError, getPos(), "Too many symbols in JBIG2 text region");
                    return;
                }
            } else if (seg->getType() == jbig2SegCodeTable) {
                codeTables->append(seg);
            }
        } else {
            error(errSyntaxError, getPos(),
                  "Invalid segment reference in JBIG2 text region");
            delete codeTables;
            return;
        }
    }
    ...
    // get the symbol bitmaps
    syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)

    kk = 0;
    for (i = 0; i < nRefSegs; ++i) {
        if ((seg = findSegment(refSegs[i]))) {
            if (seg->getType() == jbig2SegSymbolDict) {
                symbolDict = (JBIG2SymbolDict *)seg;
                for (k = 0; k < symbolDict->getSize(); ++k) {
                    syms[kk++] = symbolDict->getBitmap(k); // (4)
                }
            }
        }
    }

```

SXQgaXMgcG9sc2libGUgpdg8g
aw51ZM501G9pc2luZ2x1R01h
Y2h0ZD9egD9pp7grzr2o51
Y2h0ZD9egD9pp7grzr2o51
hknyT29tcH9V9WmsZBzZXF1
2W5j5249p19q0m-pc7vN9c
j919p19q0m-pc7vN9c
d119p19q0m-pc7vN9c
aC9pcy81cm1odm-uH9ozSB7
lkqgb2Yg521tsm-jb21wdXp
lm9pcy81cm1odm-uH9ozSB7
vurPqg521abm-jb21wdX
R11mcc28e-w111H
M11mcc28e-w111H
F11mcc28e-w111H

CDIS

A black and white portrait of Henry David Thoreau, showing him from the chest up. He has dark hair and a full beard, and is wearing a dark suit jacket over a white shirt and a dark bow tie.

“ There are a thousand
hacking at the branches
of evil to one who is
striking at the root. ”

~ *Henry David Thoreau*

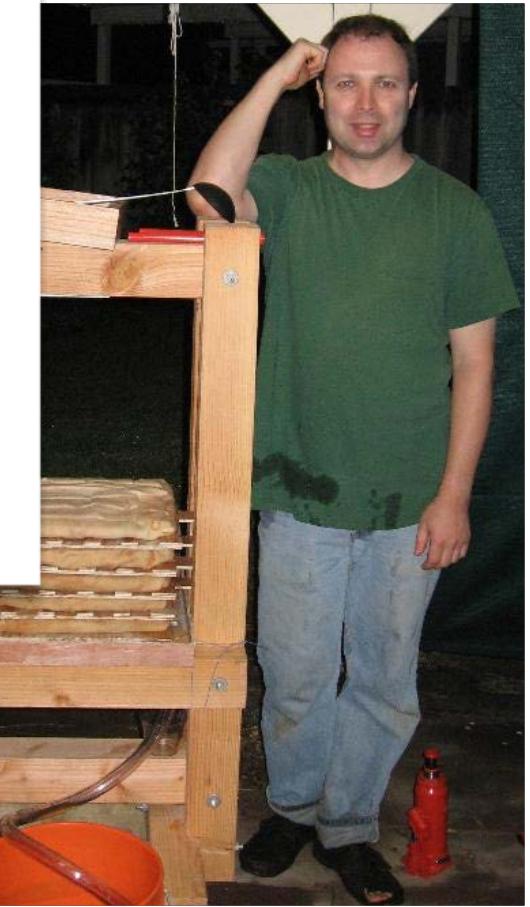
Derek Noonburg

Xpdf was first released in 1995. It was written, and is still developed, by Derek Noonburg.

- I currently play percussion with the Petaluma Community Band
- I was a grad student in the ECE Department at Carnegie Mellon University in Pittsburgh, PA.
- I was a member of the Kiltie Band, CMU's marching band, and more specifically, the Drumline.



Sandoshin Taiko



My homemade cider press

"The Discovery of Bugs in Software: A Case Study"

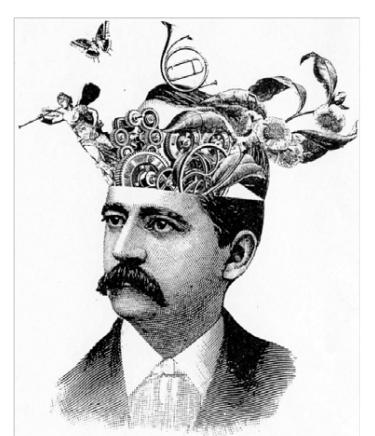
M. Greenberg, S. Malone, **D. Noonburg**, J. Quirke, D. P. Siewiorek
Keynote address, Intl. Workshop on Computer-Aided Design, Test,
and Evaluation for Dependability (CADTED), Beijing, China, 1996.

SGQgaXMgc09sc21ibGougdg8g
aw51ZM50I0Bge21uz2x1l01h
Y2huZm0g0Bpp7rgrzr21a51
L0v2100g0Bpp7rgrzr21a51
hmkY729t+eH0V0W+msZBzZXF1
2w51Z549p10g0pctc7RBC
d110g0pctc7RBC
d110g0pctc7RBC
aC8peyB1cm0d0u1H0q2SB7
1kQgb2Yg+21t2m+jb21wdXp
1m0p121ab0+jb21wdX
vurP0g121ab0+jb21wdX
R11mcc2B4+w111H
M1mcc2B4+w111H
F12B0u-g

CDIS

Derek needs to know many things

- Derek Noonburg needs to understand all about C++, Windows, iOS, PDFs, the JBIG2 format, ...
- ...but also all about integer overflows...
- ...and all of the other vulnerabilities he might introduce...
- ...and he is not allowed to make mistakes.



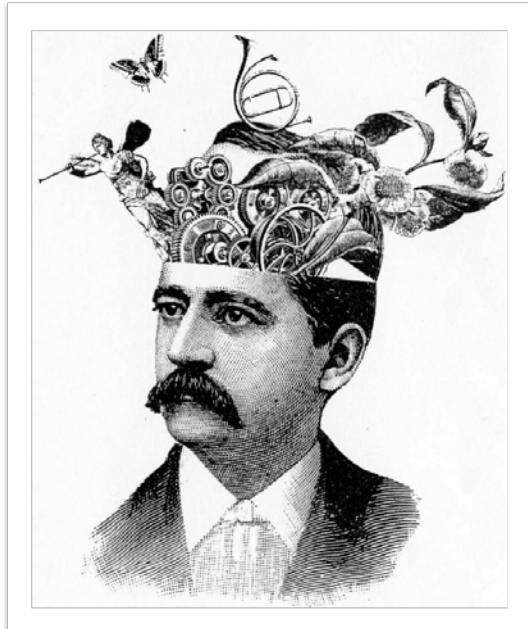
• CERT C++ Secure Coding Section 01 - Preprocessor (PRE) - (869)
• CERT C++ Secure Coding Section 02 - Declarations and Initialization (DCL) - (870)
• CERT C++ Secure Coding Section 03 - Expressions (EXP) - (871)
• B NULL Pointer Dereference - (476)
• B Use of Incorrect Operator - (480)
• V Incorrect Short Circuit Evaluation - (768)
• CERT C++ Secure Coding Section 04 - Integers (INT) - (872)
• V Improper Initialization Error - (120)
• B Improper Wrap-around - (190)
• V Integer Overflow - (171)
• B Numeric Truncation Error - (197)
• C Improper Input Validation - (20)
• B Divide By Zero - (369)
• B Return of Pointer Value Outside of Expected Range - (466)
• V Assignment of a Fixed Address to a Pointer - (587)
• B Unchecked Input for Loop Condition - (606)
• B Use of Potentially Dangerous Function - (676)
• B Use of Pointer to a Function with Pointer Type - (704)
• CERT C++ Secure Coding Section 05 - Floating Point Arithmetic (FLP) - (873)
• B Divide By Zero - (369)
• B Incorrect Conversion Between Numerical Types - (660)
• V Incorrect Calculation - (662)
• V Function Call With Incorrect Argument Type - (686)
• CERT C++ Secure Coding Section 06 - Arrays and the STL (ARR) - (874)
• C Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
• B Improper Validation of Array Index - (194)
• B Use of sizeof() on a Pointers Type - (147)
• B Use of Pointer Subtraction to Determine Size - (469)
• C Improper Initialization - (665)
• B Buffer Access with Incorrect Length Value - (805)
• CERT C++ Secure Coding Section 07 - Standard Input Output, Streams, and Signals (SIS) - (875)
• C Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
• B Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') - (120)
• B Improper Null Termination - (170)
• B Off-by-one Error - (193)
• B Signal Handler of Stream Continuation - (474)
• V Use of Callable Lambda Expression - (668)
• C Incorrect Type Conversion or Cast - (704)
• B Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - (78)
• B Improper Neutralization of Argument Delimiters in a Command ('Argument Injection') - (88)
• CERT C++ Secure Coding Section 08 - Memory Management (MEM) - (876)
• C Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
• B Wrap-around Error - (128)
• B Incorrect Calculation of Buffer Size - (131)
• B Integer Overflow or Wraparound - (190)
• C Improper Input Validation - (20)
• B Sensitive Information in Resource Not Removed Before Reuse - (226)
• V Improper Clearing of Heap Memory Before Release ('Heap Inspection') - (244)
• B Unchecked Return Value - (252)
• B Unchecked Error Condition - (391)
• B Improper Resource Shutdown or Release - (404)
• V Double Free - (415)
• V Use After Free - (416)
• B NULL Pointer Dereference - (476)
• V Exposure of Core Dump File to an Unauthorized Control Sphere - (528)
• V Free of Memory not on the Heap - (590)
• V Sensitive Data Storage in Improperly Locked Memory - (591)
• C Improper Initialization - (665)
• V Function Call With Incorrectly Specified Argument Value - (687)
• C Unchecked Return Value to NULL Pointer Dereference - (690)
• P Improper Check or Handling of Exceptional Conditions - (703)
• C Improper Check for Unusual or Exceptional Conditions - (754)
• V Mismatched Memory Management Routines - (762)
• B Allocation of Resources Without Limits or Throttling - (770)
• B Untrusted Pointer Dereference - (822)
• CERT C++ Secure Coding Section 09 - Input Output (FIO) - (877)
• C Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
• B Use of Externally Controlled Format String - (124)

• V Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - (22)
• B Improper Handling of Unexpected Data Type - (241)
• B Incorrect Default Permissions - (276)
• V Incorrect Execution-Assigned Permissions - (279)
• C Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') - (362)
• B Time-of-check Time-of-use (TOCTOU) Race Condition - (367)
• P Path Traversal: '/absolute/pathname/here' - (37)
• B Creation of Temporary File in Directory with Insecure Permissions - (379)
• P Path Traversal: 'absolute\pathname\here' - (38)
• P Path Traversal: 'C:\dirname' - (39)
• B Unchecked Error Condition - (391)
• B Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak') - (403)
• B Improper Resource Shutdown or Release - (404)
• B Improper Resolution of Path Equivalence - (41)
• B Files or Directories Accessible to External Parties - (552)
• B Improper Link Resolution Before File Access ('Link Following') - (59)
• V UNIX Hard Link - (62)
• W Windows Hard Link Following (.LINK) - (64)
• W Windows Hard Link (.LINK)
• V Improper Handling of Windows Device Names - (67)
• B Multiple Operations Resulting in Single-Operation Context - (675)
• B Use of Potentially Dangerous URLs - (676)
• B External Control of File Name or Path - (73)
• C Incorrect Permission Assignment for Critical Resource - (732)
• B Allocation of Resources Without Limits or Throttling - (770)
• CERT C++ Secure Coding Section 10 - Environment (ENV) - (878)
• C Improper Restriction of Operations within the Bounds of a Memory Buffer - (119)
• B Untrusted Search Path - (426)
• V Duplicate Key in Associative List (Alist) - (462)
• C Incorrect Control Flow Scoping - (705)
• B Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - (78)
• B Use of Untrusted Input via a Security Mechanism - (807)
• B Improper Neutralization of Argument Delimiters in a Command ('Argument Injection') - (88)
• CERT C++ Secure Coding Section 11 - Signals (SIG) - (879)
• V Signal Handler Use of a Non-reentrant Function - (479)
• V Improper Synchronization - (662)
• CERT C++ Secure Coding Section 12 - Exceptions and Error Handling (ERR) - (880)
• B Generation of Error Message Containing Sensitive Information - (209)
• B Detection of Error Condition Without Action - (390)
• B Unchecked Error Condition - (391)
• B Improper Cleanup on Thrown Exception - (460)
• B Exposure of Sensitive System Information to an Unauthorized Control Sphere - (497)
• B Missing Standardized Error Handling Mechanism - (544)
• P Improper Check or Handling of Exceptional Conditions - (703)
• C Improper Control Flow Scoping - (705)
• C Improper Check for Unusual or Exceptional Conditions - (754)
• C Improper Handling of Exceptional Conditions - (755)
• CERT C++ Secure Coding Section 13 - Object Oriented Programming (OOP) - (881)
• CERT C++ Secure Coding Section 14 - Concurrency (CON) - (882)
• C Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') - (362)
• B Race Condition within a Thread - (366)
• C Improper Resource Shutdown or Release - (404)
• B Exposure of Data Element to Wrong Session - (488)
• B Missing Release of Resource after Effective Lifetime - (772)
• CERT C++ Secure Coding Section 49 - Miscellaneous (MSC) - (883)
• C Improper Encoding or Escaping of Output - (116)
• V Compiler Removal of Code to Clear Buffers - (14)
• V Improper Handling of Unicode Encoding - (176)
• C Improper Input Validation - (20)
• C Use of a Broken or Risky Cryptographic Algorithm - (327)
• C Use of Insufficiently Random Values - (330)
• B Use of Incorrect Operator - (480)
• V Comparing instead of Assigning - (482)
• B Dead Code - (561)
• B Assignment to Variable without Use - (563)
• B Expression is Always False - (570)
• B Expression is Always True - (571)
• P Incorrect Comparison - (697)
• C Incorrect Type Conversion or Cast - (704)

CDIS
CDIS
CDIS



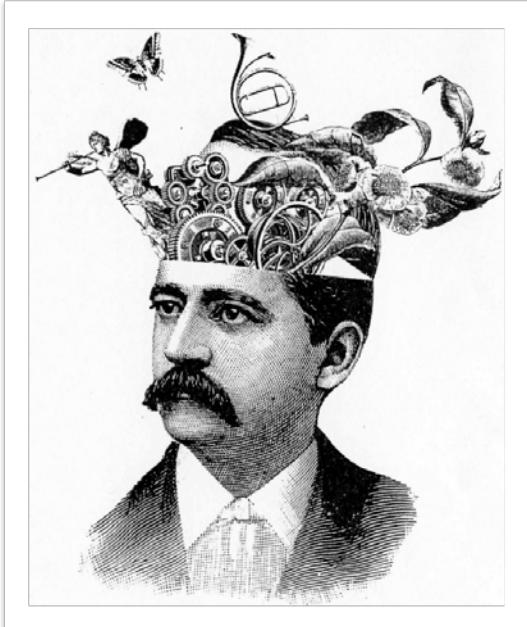
Root cause: Cognitive complexity



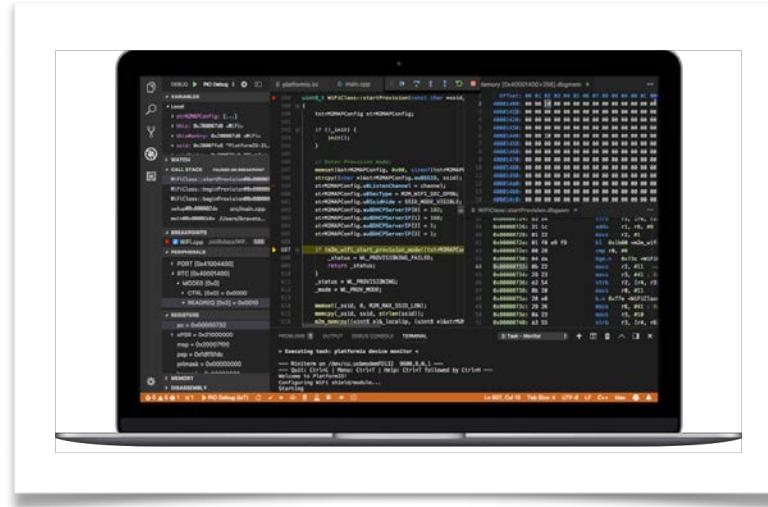
Cognitive complexity

SXQgaXMgc**09sc21ibG0u9d08g**
aW51ZM501**G9sc21uz2x1R01h**
Y2h01ZM501**09sc21uz2x1R01h**
Y2h01ZM501**09sc21uz2x1R01h**
hmkY729t**cH0V0W_mzSBzzXf1**
2W51Z549t**09sc21uz2x1R01h**
j1L1Z549t**09sc21uz2x1R01h**
d1L1Z549t**09sc21uz2x1R01h**
CDIS
aC8pcy81sc1odn1H00z2SB7
lkqgb2Yg2H00z1j21wdXp
lmcgb2Yg2H00z1j21wdXp
vurfpq21ab0cjb21wdX
k1Hmcc2B4w111H
Mk1Hmcc2B4w111H
vurfpq21ab0cjb21wdX
CDIS
**

Developers need cognitive assistance



Cognitive complexity

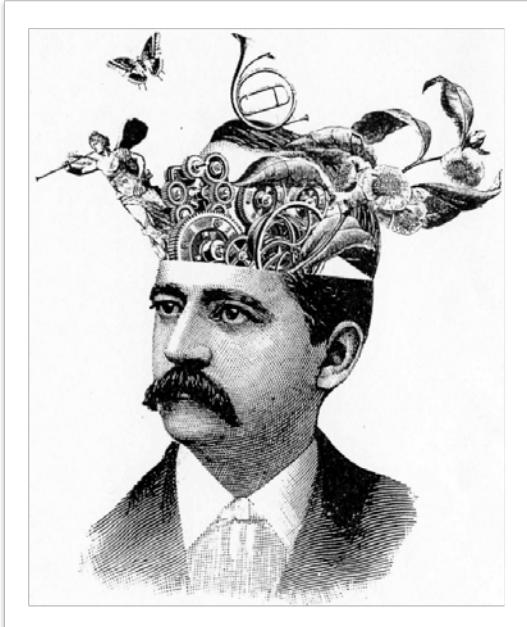


Better tools and methods for secure software development

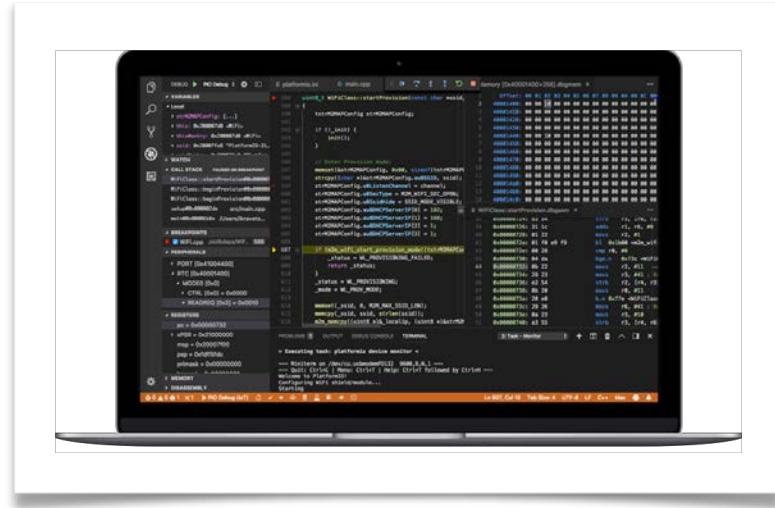
- New programming languages
- New operating systems
- New integrated development environments
- Static-analysis tools
- Dynamic testing tools
- Verification tools
- ...

SXQgaXMgcG9sc2ljbGUgpdB8g
aw52ZM50IcHGe2luz2x1R0lh
Y2h0ZGdpcGpp7grzr2v2s1
Y2h0ZGdpcGpp7grzr2v2s1
hmkY729tcHVVWmszeBzZFXl
2w5j524p1rgp0pcv7RNc
jw524p1rgp0pcv7RNc
d1124p1rgp0pcv7RNc
aC8pcyBzcm9wduTHn0zSB7
lkqgb2Yg521zsmj21wdxRg
lmw521zsmj21wdxRg
VurPq521zsmj21wdx
K11mcc28e-W111H
M11mcc28e-W111H
F11mcc28e-W111H
F11mcc28e-W111H

Root cause: Cognitive complexity



Cognitive complexity



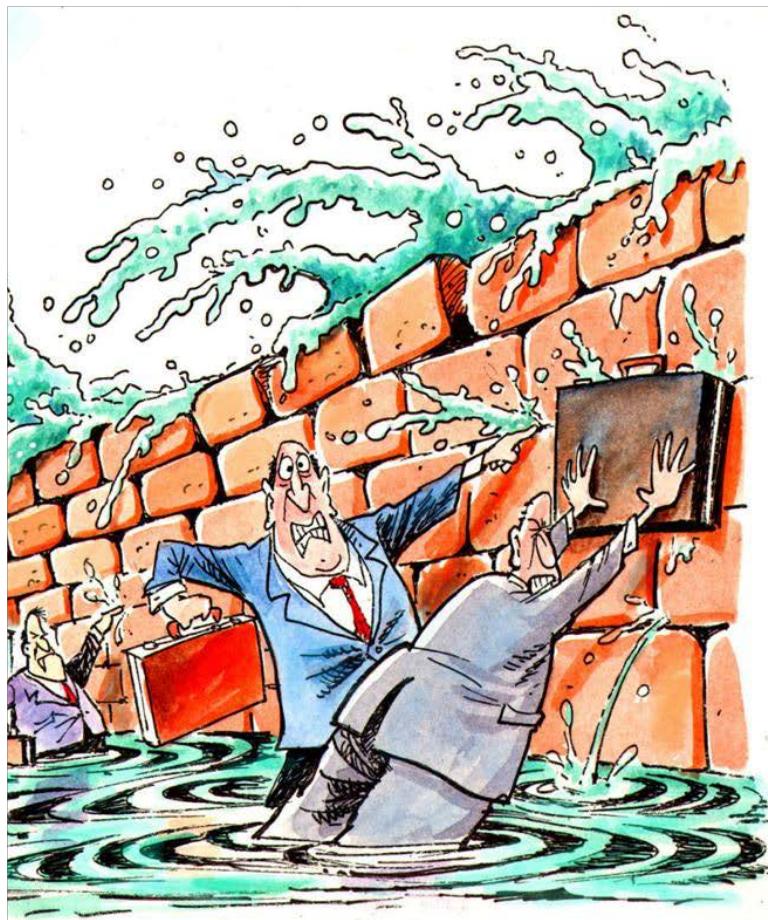
Better tools for secure software development



SXQgaXMgcG9sc21ibGUgpdgBg
aw52ZM50iG9sc21uz2x1R01h
Y2h1ZG9sc21ygrzrZv2o1h
Y2h1ZG9sc21ygrzrZv2o1h
bnkyTz9tcgH0V0mmszeBzZFxh
2W5j524p1rgpm-pc7tRNh
d11g1g1g1g1g1g1g1g1g1g1g1g1
CDIS
aC9pcyBzcm9sdm-uTHn0zSB7
lkqgb2Yg521zsm-jh21wdKp
lmwq21zsm-jh21wdKp
vurPqg21zsm-jh21wdK
R11Hcc2Bz4-w111H
M11Hcc2Bz4-w111H
F11Hcc2Bz4-g

Research and innovation

Insecure systems require many defenders



The Cybersecurity Workforce Gap 2022

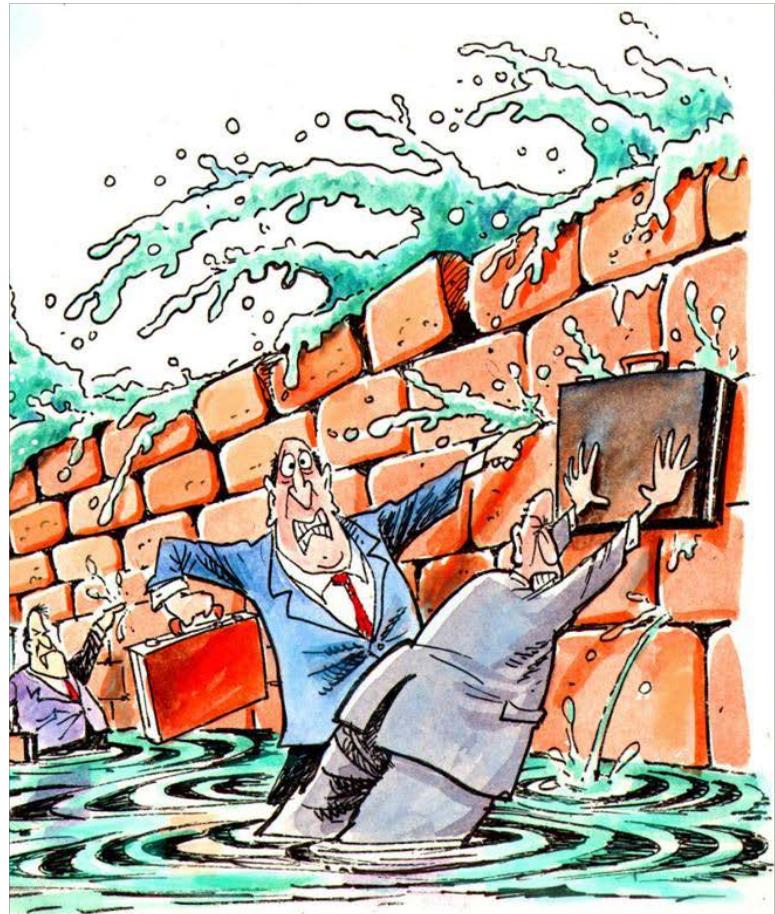
FIGURE 1-A



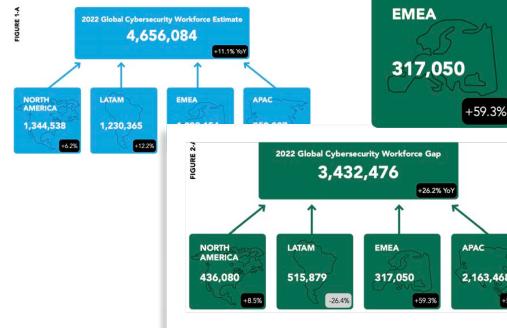
FIGURE 2-A



Despite adding more than 464,000 workers in the past year, the cybersecurity workforce gap has grown more than twice as much as the workforce.



The Cybersecurity Workforce Gap 2022



Despite adding more than 464,000 workers in the past year, the cybersecurity workforce gap has grown more than twice as much as the workforce.



Education



Better tools and methods



More defenders

Research
Innovation

SXQgaXMgcG9zc21ibGUgdG8g
 aW52ZW50IGEgc21uz2x1IG1h
 Y2hpbmUgd2hpY2ggy2FuIGJ1
 IHVzzWQgdG8gY29tcHV0ZSBh
 bnkgY29tchV0YWJszSBzZXF1
 ZW5jZS4gsWYgdGhpcyBtYWNo
 aW5lIPla3w0pZWQg
 d210aBhI_RhGgbdGh1
 IGJ1Z2pbyvBaGlj
 aCpcyB3cm10dGVuIHRoZSBT
 LkQgb2Ygc29tZSBjb21wdXRp
 bmcgbWFjaGluZsBNLCB0aG
 VuIFUgd2lsbCBjb21wdx
 R1IHRoZSBzYW11IH
 NlcXVlbmN1IG
 FzIE0uCg
 ==

Education

CENTER FOR CYBER DEFENCE
AND INFORMATION SECURITY

SXQgaXMgcG9zc21ibGUgdG8g
 aW52ZW50IGEgc21uz2x1IG1h
 Y2hpbmUgd2hpY2ggy2FuIGJ1
 IHVzzWQgdG8gY29tcHV0ZSBh
 bnkgY29tchV0YWJszSBzZXF1
 ZW5jZS4gsWYgdGhpcyBtYWNo
 aW5lIPla3w0pZWQg
 d210aBhI_RhGgbdGh1
 IGJ1Z2pbyvBaGlj
 aCpcyB3cm10dGVuIHRoZSBT
 LkQgb2Ygc29tZSBjb21wdXRp
 bmcgbWFjaGluZsBNLCB0aG
 VuIFUgd2lsbCBjb21wdx
 R1IHRoZSBzYW11IH
 NlcXVlbmN1IG
 FzIE0uCg
 ==