Coverage-guided fuzzing using LLVM on Postgres code to find security issues in database functions and operators.

Or….

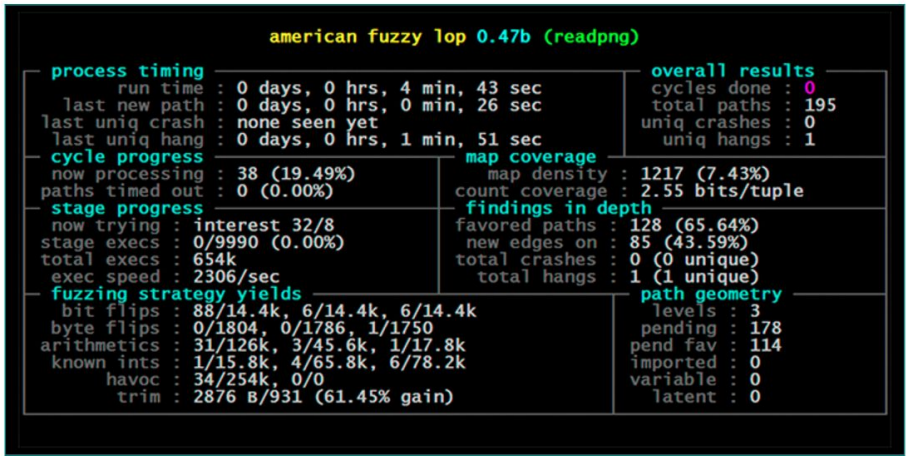
What I did for fun during my summer vacation!

What is fuzzing?

lcamtuf.coredump.cx/afl/

american fuzzy lop (1.94b)

American fuzzy lop is a security-oriented fuzzer that employs a novel type of compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states in the targeted binary. This substantially improves the functional coverage for the fuzzed code. The compact synthesized corpora produced by the tool are also useful for seeding other, more labor- or resource-intensive testing regimes down the road.



Issues Fuzzing Postgres

Most open source fuzzers

- Expect to exec a binary repeatedly on text inputs
- Require modifying C source to call function being tested
- Postgres bugs rarely cause crashes but get caught and signal unexpected errors

Ideally we want

- Not to have to modify the client/server architecture of Postgres
- A generic function that can be provided an expression to evaluate repeatedly
- And a harness that understands which errors are expected or unexpected

LLVM Libfuzzer

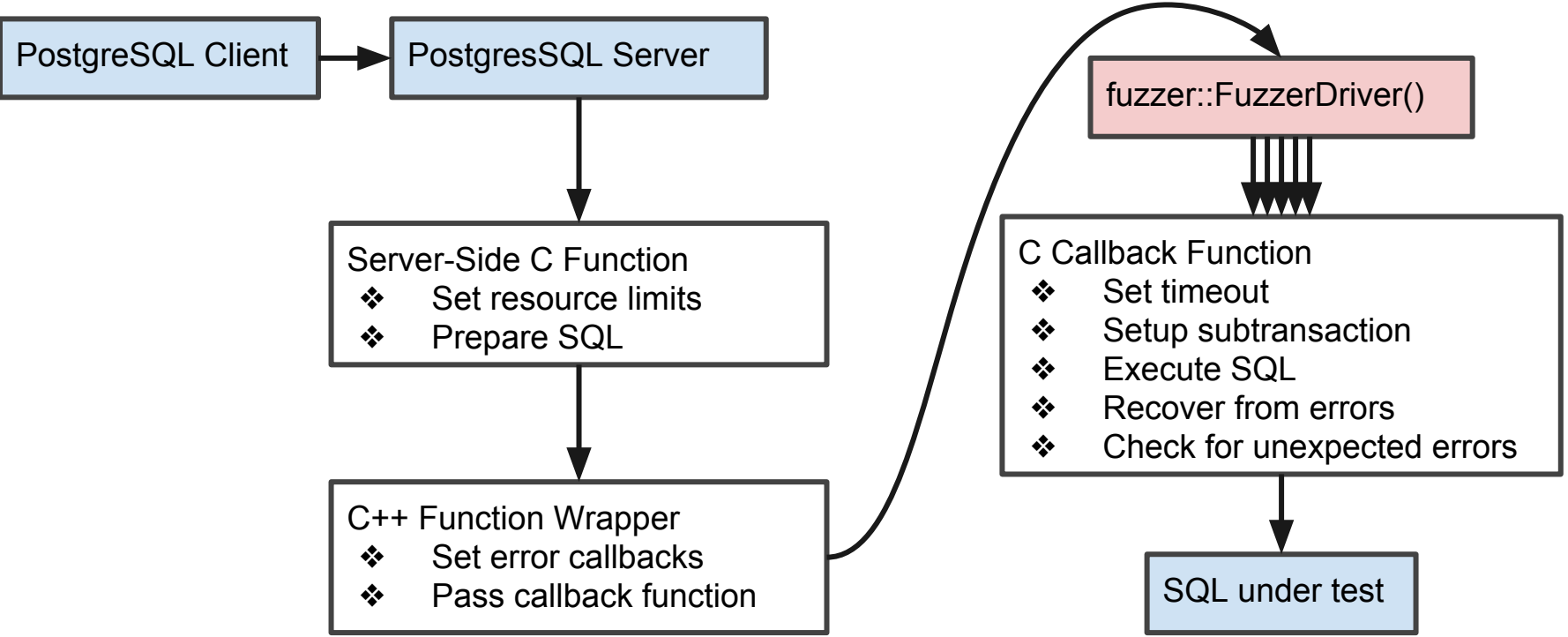
Pros

- In-process so we can call it from the server
- Very very fast (no syscalls at all, coverage data is in local memory)
- Flexible set of tools that we can pick the parts to keep or reimplement

Cons

- Immature -- not as clever at generating test cases as AFL
- Rough edges -- intended to be used in LLVM’s own builds

Flow of Control



Server-Side C Function

- Set resource limits
  - Core files off (Libfuzzer calls abort())
  - CPU 1s soft (SIGXCPU) *
  - CPU 5min hard (SIGKILL)
  - Memory 200MB
- Prepare SQL

```
SPIPlanPtr plan;

PG_FUNCTION_INFO_V1(fuzz);
Datum
fuzz(PG_FUNCTION_ARGS)
{
    unsigned runs = PG_GETARG_INT32(0);
    text *expr_text = PG_GETARG_TEXT_P(1);
    char *expr = text_to_cstring(expr_text);
    Oid argtypes[1] = { TEXTOID };

    struct rlimit new;
    new.rlim_cur = new.rlim_max = 0;
    setrlimit(RLIMIT_CORE, &new);
    new.rlim_cur = 1; new.rlim_max = 300;
    setrlimit(RLIMIT_CPU, &new);
    new.rlim_cur = 200000000; new.rlim_max = RLIM_INFINITY;
    setrlimit(RLIMIT_DATA);

    SPI_connect();
    /* Prepare once before we start the driver */
    plan = SPI_prepare(expr, 1, argtypes);
    /* Invoke the driver via the C++ code */
    GoFuzz(runs);
    SPI_finish();

    PG_RETURN_NULL();
}
```

C++ Wrapper Function

- Set error callbacks
  - aborthandler
  - staticdeathcallback
  - staticerrorcallback
- Pass callback function
  - FuzzOne

```
int GoFuzz(unsigned runs) {
    char runarg[] = "-runs=400000000999";
    sprintf(runarg, "-runs=%u", runs);
    char *argv[] = {
        "PostgresFuzzer",
        runarg,
        "-verbosity=1",
        "-only_ascii=1",
        "-timeout=60",
        "-report_slow_units=1",
        "-use_traces=1",
        "/var/tmp/corpus",
        "-max_len=32",
        NULL
    };
    int argc = sizeof(argv)/sizeof(*argv) - 1;

    /* Catch abort and print out the test case */
    struct sigaction sigact;
    memset(&sigact, 0, sizeof(sigact));
    sigact.sa_sigaction = aborthandler;
    sigaction(SIGABRT, &sigact, 0);

    return fuzzer::FuzzerDriver(argc, argv, FuzzOne);
}
```

C Callback Function (1)

- Set timeout
  - 100ms (varied over tests)
  - Will need to be a parameter
- Setup subtransaction
  - BeginInternalSubTransaction
  - Save Memory Context
  - Save Resource Owner
- Execute SQL
  - SPI_execute_plan()

```
void FuzzOne(const char *Data, size_t Size) {
    text *arg = cstring_to_text_with_len(Data, Size);

    MemoryContext oldcontext = CurrentMemoryContext;
    ResourceOwner oldowner = CurrentResourceOwner;

    BeginInternalSubTransaction(NULL);
    PG_TRY();
    {
        Datum values[1] = { PointerGetDatum(arg) };
        int retval;

        /* Slow queries are bad but if they CHECK_FOR_INTERRUPTS often
           enough then that's not too bad. We must directly call
           enable_timeout because STATEMENT_TIMEOUT is only armed in
           postgres.c which SPI bypasses */
        CHECK_FOR_INTERRUPTS();
        enable_timeout_after(STATEMENT_TIMEOUT, 100);

        retval = SPI_execute_plan(plan, values,
                                  NULL /* nulls */,
                                  true, /* read-only */
                                  0 /* max rows */);
        disable_timeout(STATEMENT_TIMEOUT, true);

        SPI_freetuptable(SPI_tuptable);

        ReleaseCurrentSubTransaction();
        MemoryContextSwitchTo(oldcontext);
        CurrentResourceOwner = oldowner;
        SPI_restore_connection();
    }
```

C Callback Function (2)

- Recover from errors
  - Restore Memory Context
  - Restore Resource Owner
  - Set aside ErrorData
  - Roll Back Subtransaction

```
PG_CATCH();
{
    /* Save error info */
    MemoryContextSwitchTo(oldcontext);
    disable_timeout(STATEMENT_TIMEOUT, true);

    ErrorData  *edata = CopyErrorData();
    inc_errcode_count(edata->sqlerrcode);


    /* Attempt to recover using the subtransaction */
    FlushErrorState();
    /* Abort the inner transaction */
    RollbackAndReleaseCurrentSubTransaction();
    MemoryContextSwitchTo(oldcontext);
    CurrentResourceOwner = oldowner;
    SPI_restore_connection();
```

C Callback Function (3)

- Check for unexpected errors
  - Resource Limits
  - Internal Errors
  - Internal Regexp Errors
  - Call errorcallback() *

```
    /* INTERNAL_ERROR is definitely a bug. The others debatable but in
     * particular we're interested in infinite recursion caught by
     * check_for_stack_depth() which shows up as STATEMENT_TOO_COMPLEX
     * which is in the PROGRAM_LIMIT_EXCEEDED category
     */
    int errcategory = ERRCODE_TO_CATEGORY(edata->sqlerrcode);
    int regerrcode = ... // elided for space

    if (errcategory == ERRCODE_PROGRAM_LIMIT_EXCEEDED ||
        errcategory == ERRCODE_INSUFFICIENT_RESOURCES ||
        errcategory == ERRCODE_INTERNAL_ERROR ||
        (edata->sqlerrcode == ERRCODE_INVALID_REGULAR_EXPRESSION &&
         (regerrcode == REG_ESPACE || regerrcode == REG_ASSERT ||
          regerrcode == REG_INVARG || regerrcode == REG_MIXED  ||
          regerrcode == REG_ECOLORS)))
        {
            char errorname[80];
            sprintf(errorname, "error-%s", unpack_sql_state(edata->sqlerrcode));
            fprintf(stderr, "Calling errocallback for %s (%s)\n",
                    errorname, edata->message);
            errorcallback(errorname);
            FreeErrorData(edata);
        }
    else
        FreeErrorData(edata);
}
PG_END_TRY();
```

CREATE FUNCTION fuzz()

```
stark=> CREATE FUNCTION fuzz(integer,text)
    RETURNS text LANGUAGE C
    AS '/home/stark/src/libfuzzer-pg/fuzz.so','fuzz';
CREATE FUNCTION


stark=> select fuzz(1000000,'select $1::timestamptz');
...
```

CREATE FUNCTION fuzz()

```
stark=> select fuzz(1000000,'select $1::timestamptz');
#0      READ   cov: 0 bits: 0 units: 590 exec/s: 0
#1      pulse  cov: 10739 bits: 2999 units: 590 exec/s: 0
#2      pulse  cov: 10745 bits: 3080 units: 590 exec/s: 0
#4      pulse  cov: 10816 bits: 3273 units: 590 exec/s: 0
#8      pulse  cov: 10918 bits: 4005 units: 590 exec/s: 0
#16     pulse  cov: 11335 bits: 4691 units: 590 exec/s: 0
#32     pulse  cov: 11435 bits: 4950 units: 590 exec/s: 0
#64     pulse  cov: 11642 bits: 5636 units: 590 exec/s: 0
#256    pulse  cov: 11955 bits: 7099 units: 590 exec/s: 0
#512    pulse  cov: 12003 bits: 7661 units: 590 exec/s: 0
#590    INITED cov: 12005 bits: 7679 units: 315 exec/s: 0
```

CREATE FUNCTION fuzz()

```
#1024 pulse  cov: 12005 bits: 7679 units: 315 exec/s: 0
#2048 pulse  cov: 12005 bits: 7679 units: 315 exec/s: 0
#4096 pulse  cov: 12005 bits: 7679 units: 315 exec/s: 0
#8192 pulse  cov: 12005 bits: 7679 units: 315 exec/s: 8192
#16384        pulse  cov: 12005 bits: 7679 units: 315 exec/s: 8192
#18442        NEW    cov: 12005 bits: 7681 units: 316 exec/s: 6147 L: 25 m^^(nut[^( ^[*Sj-0[-e9[sf
#20574        NEW    cov: 12005 bits: 7682 units: 317 exec/s: 6858 L: 27 9::mYT+1S'0::  6:; ;09:: 6:
#32768        pulse  cov: 12005 bits: 7682 units: 317 exec/s: 6553
#34643        NEW    cov: 12012 bits: 7688 units: 318 exec/s: 6928 L: 12 j'Roc  6b6G
#35409        NEW    cov: 12012 bits: 7689 units: 319 exec/s: 7081 L: 16 j'6 G1'u97 .041
#60297        NEW    cov: 12012 bits: 7690 units: 320 exec/s: 6029 L: 27 iilliseu YYcY-:Y: -F-Y-s:-o
#63284        NEW    cov: 12012 bits: 7691 units: 321 exec/s: 6328 L: 30 'j6 Gq68C9%*F96YC. 24: "(  Pm
#64476        NEW    cov: 12012 bits: 7706 units: 322 exec/s: 6447 L: 19 (8;(\xc;9.9PYST1(.PDT
```

CREATE FUNCTION fuzz()

```
#65536      pulse  cov: 12012 bits: 7706 units: 322 exec/s: 6553
#70689      NEW    cov: 12012 bits: 7709 units: 323 exec/s: 6426 L: 15 d8!6uP9uJ91 YX6
#72861      NEW    cov: 12012 bits: 7710 units: 324 exec/s: 6071 L: 25 j 86'j) 199GET70710  [h61
#102491     NEW    cov: 12012 bits: 7713 units: 325 exec/s: 6405 L: 29 !(za011[-0(-9?-0-27 :+10:0:21
#112248     NEW    cov: 12012 bits: 7716 units: 326 exec/s: 6236 L: 26 Y0\xad8!2 6+10+F0Y0+F5P)9n'
#116686     NEW    cov: 12012 bits: 7719 units: 327 exec/s: 6482 L: 21 y](660)  m' 6 (LHdT(8
#131072     pulse  cov: 12012 bits: 7719 units: 327 exec/s: 6241
#131531     NEW    cov: 12012 bits: 7722 units: 328 exec/s: 6263 L: 29 *y]u(7:60(11-03-860986696!62;
#135066     NEW    cov: 12012 bits: 7723 units: 329 exec/s: 6431 L: 32 9:.40:2000-03-15 12:14:039L G;MC
#141231     NEW    cov: 12012 bits: 7726 units: 330 exec/s: 6419 L: 26 ,;; j6 qGtomo68Crrow EST9'
#189651     NEW    cov: 12012 bits: 7736 units: 331 exec/s: 6321 L: 12 j6 Gqud68C9%
#191339     NEW    cov: 12012 bits: 7738 units: 332 exec/s: 6377 L: 26 201-10-32720 02*01-09-2r00
#197481     NEW    cov: 12012 bits: 7741 units: 333 exec/s: 6171 L: 32  j6  PmGqd8Cmill9%*Fise9co6YXF.j
```

CREATE FUNCTION fuzz()

```
#205513     NEW    cov: 12012 bits: 7742 units: 334 exec/s: 6227 L: 29  j6  oPeGqd8C9%*FF96YaXFm'-F.
#218191     NEW    cov: 12012 bits: 7743 units: 335 exec/s: 6234 L: 30 Yoon//Fe"-aq1on/o0\xa2n/ (o/ ]bc
#224296     NEW    cov: 12012 bits: 7746 units: 336 exec/s: 6230 L: 22 2(8;(\xc;9.1(00.0-0#-15
#237337     NEW    cov: 12012 bits: 7747 units: 337 exec/s: 6085 L: 30 Y'Y*j,3()"Y\xa;:, Pm'*H(6*y (T]
#239500     NEW    cov: 12012 bits: 7750 units: 338 exec/s: 6141 L: 22 d6 `6!9uPj9J( eDOY966(
#262144     pulse  cov: 12012 bits: 7750 units: 338 exec/s: 6096
#267757     NEW    cov: 12012 bits: 7751 units: 339 exec/s: 6085 L: 32 (6(9-9*F96YX.iFjYN0)Y-:YF-Y-::Y6
#297841     NEW    cov: 12013 bits: 7752 units: 340 exec/s: 6078 L: 24 'Y(j,: 9PST8PFT/Y(dst;(!
#303901     NEW    cov: 12013 bits: 7753 units: 341 exec/s: 6078 L: 30 C8aM^(\!d6Lu9X\Y9  Rj6 o 1(P6m
#305896     NEW    cov: 12013 bits: 7754 units: 342 exec/s: 6117 L: 23 9:mY7.^;;('Tt+.7.t+J.7~
#392276     NEW    cov: 12013 bits: 7755 units: 343 exec/s: 6226 L: 28 '  6::.9 6::.197.041 (mq7:3F
#407931     NEW    cov: 12013 bits: 7756 units: 344 exec/s: 6180 L: 27 -infinity Pm'H^1^(\xb+Hoo8/($
#447063     NEW    cov: 12013 bits: 7758 units: 345 exec/s: 6209 L: 31 0 1-03-27 z 10:0:000000799:2::1
```

CREATE FUNCTION fuzz()

```
#450949      NEW    cov: 12013 bits: 7760 units: 346 exec/s: 6177 L: 27 2011-020-271-009019970221::
#476579      NEW    cov: 12013 bits: 7761 units: 347 exec/s: 6270 L: 28 (7 *\M\Z.`()(J.7^:e.;,Yo;o./
#489858      NEW    cov: 12013 bits: 7762 units: 348 exec/s: 6200 L: 32 j6F0F'YYYto*j,: S0F5l(daFy9Y(F6;
#490971      NEW    cov: 12013 bits: 7763 units: 349 exec/s: 6214 L: 30 j6F0GYY'Y*j,: F0FYl(F9Y(F6(;l9
#501972      NEW    cov: 12013 bits: 7765 units: 350 exec/s: 6197 L: 26 j6AFmerim/cAesica/DoNj6 Ge
#506861      NEW    cov: 12013 bits: 7770 units: 351 exec/s: 6257 L: 32 19 [-6'7.041  06::.9 6::.8AFmqF7
#507341      NEW    cov: 12013 bits: 7771 units: 352 exec/s: 6263 L: 31 6 ,F:6 ,6F ,!F:6 ,F:-9.N:-9..:N
#509461      NEW    cov: 12013 bits: 7772 units: 353 exec/s: 6212 L: 30 197.041 q7:3_+10:9!CLSTYST2F:0
#517081      NEW    cov: 12013 bits: 7774 units: 354 exec/s: 6229 L: 31 am'-FGFF8GFYJF -HH24-.-text-FYF
#524288      pulse  cov: 12013 bits: 7774 units: 354 exec/s: 6241
#556354      NEW    cov: 12013 bits: 7775 units: 355 exec/s: 6251 L: 27 201-#0-3720 02*01-0;&((d9-
#574472      NEW    cov: 12013 bits: 7776 units: 356 exec/s: 6244 L: 27 9:MY7(7 .^;;('.+Tt7*.t\+J.M
#580776      NEW    cov: 12013 bits: 7777 units: 357 exec/s: 6244 L: 26 2011-020-071-00902011-0199
```

CREATE FUNCTION fuzz()

```
#581747        NEW    cov: 12013 bits: 7779 units: 358 exec/s: 6255 L: 27 2011-020-2\1-00190997022201
#603216        NEW    cov: 12013 bits: 7780 units: 359 exec/s: 6283 L: 32 2011-020-2\1-001909970222020011-
#731141        NEW    cov: 12013 bits: 7781 units: 360 exec/s: 6249 L: 30 Y0\xad8!2;5Pd8!6u9 P9JXYuJ91 YX6
#832356        NEW    cov: 12013 bits: 7782 units: 361 exec/s: 6258 L: 28  j6  Pj6AFmGqd8'merim/C9cA%*
#855035        NEW    cov: 12013 bits: 7783 units: 362 exec/s: 6287 L: 22 !*j ,9F-74.Y6 J-8j9:0J
#867736        NEW    cov: 12013 bits: 7784 units: 363 exec/s: 6287 L: 30 tdoay(?7.?(7.(( J8- J8-)Y,;( '
#926231        NEW    cov: 12013 bits: 7785 units: 364 exec/s: 6300 L: 31 (?7.(?7.(([197.09-a^z(0=9.4a_1
#938266        NEW    cov: 12013 bits: 7787 units: 365 exec/s: 6297 L: 30 C8!d6Lu9XY9 Y9:m8!d6uP9 Rj6 oT
#985381        NEW    cov: 12013 bits: 7792 units: 366 exec/s: 6276 L: 30 211-002#0-3720 02*01-0;&((d1-0
Done 1000000 runs in 159 second(s)
 fuzz
------


(1 row)
```

Currently just a bit noisy...

FuzzOne n=524288 success=9896 fail=514392 null=0

Error codes seen 22007:487993 22008:12697 0A000:96 22023:10326 22009:3280

#524288 pulse cov: 12013 bits: 7774 units: 354 exec/s: 6241

LOG: could not open directory "/usr/local/pgsql/share/timezone/Zulu": Not a directory

CONTEXT: SQL statement "select $1::timestamptz"

STATEMENT: select fuzz(1000000,'select $1::timestamptz')

LOG: could not open directory "/usr/local/pgsql/share/timezone/Zulu": Not a directory

CONTEXT: SQL statement "select $1::timestamptz"

STATEMENT: select fuzz(1000000,'select $1::timestamptz')

LOG: could not open directory "/usr/local/pgsql/share/timezone/Zulu": Not a directory

CONTEXT: SQL statement "select $1::timestamptz"

STATEMENT: select fuzz(1000000,'select $1::timestamptz')

Expected errors

Error codes seen 22007:487993 22008:12697 0A000:96 22023:10326 22009:3280

Appendix A. PostgreSQL Error Codes

Error Code	Condition Name
22007	invalid_datetime_format
22008	datetime_field_overflow
22009	invalid_time_zone_displacement_value
0A000	feature_not_supported
22023	invalid_parameter_value

Unexpected errors

```
commit 258ee1b635e43a37e901fd5f62bdd5f1087d65a5
Author: Greg Stark <stark@mit.edu>
Date:   Sun Sep 6 02:04:37 2015 +0100

    Move DTK_ISODOW DTK_DOW and DTK_DOY to be type UNITS rather than
    RESERV. RESERV is meant for tokens like "now" and having them in that
    category throws errors like these when used as an input date:

    stark=# SELECT 'doy'::timestamptz;
    ERROR:  unexpected dtype 33 while parsing timestamptz "doy"
    LINE 1: SELECT 'doy'::timestamptz;
                   ^
    stark=# SELECT 'dow'::timestamptz;
    ERROR:  unexpected dtype 32 while parsing timestamptz "dow"
    LINE 1: SELECT 'dow'::timestamptz;
                   ^
```

Performance Problems

```
commit 48789c5d23a7f382e3cb721547d5e0af7a
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 16 14:14:40 2015 -0400
Fix regular-expression compiler to handle
loops of constraint arcs.

commit b63fc28776c5d2efdb4de326ad0f0b5b88
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 14:51:58 2015 -0400
Add recursion depth protections to regular
expression matching.

commit f2c4ffc3307cab6619a28e77da9211416c
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 14:26:36 2015 -0400
Fix potential infinite loop in regular
expression execution.
```

```
commit 9fe8fe9c9e5d7fc099acfc96e976ee72b2b
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 13:45:39 2015 -0400
Add some more query-cancel checks to regul
expression matching.

commit 558d4ada1851274fe4dd3618f3f6561b638
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 13:30:42 2015 -0400
Docs: add disclaimer about hazards of usin
regexps from untrusted sources.
```

Security Problems?

We'll have to wait for 9.5.1 security release to see...

Fuzzers are popular...

Other people are experimenting with fuzzers:

- Piotr Stefaniak has been running Libfuzzer too
- Andreas Seltenreich wrote sqlsmith which generates random SQL https://github.com/anse1

My Libfuzzer experimental work is at:

https://github.com/gsstark/libfuzzer-pg