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?

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
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()
CREATE FUNCTION fuzz()
CREATE FUNCTION fuzz()

<table>
<thead>
<tr>
<th>Line</th>
<th>Status</th>
<th>Coverage</th>
<th>Bits</th>
<th>Units</th>
<th>Executions/s</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>#450949</td>
<td>NEW</td>
<td>12013</td>
<td>7760</td>
<td>346</td>
<td>6177</td>
<td>27 2011-020-271-009019970221::</td>
</tr>
<tr>
<td>#476579</td>
<td>NEW</td>
<td>12013</td>
<td>7761</td>
<td>347</td>
<td>6270 L: 28 (7 *M\Z.`(J.7^:e.;,Yo;o./</td>
<td></td>
</tr>
<tr>
<td>#489858</td>
<td>NEW</td>
<td>12013</td>
<td>7762</td>
<td>348</td>
<td>6200 L: 32 j6F0F'YYYto*j,: S0F5l(daFy9Y(F6;</td>
<td></td>
</tr>
<tr>
<td>#490971</td>
<td>NEW</td>
<td>12013</td>
<td>7763</td>
<td>349</td>
<td>6214 L: 30 j6F0GYY'Y*j,: F0FYI(F9Y(F6;I9</td>
<td></td>
</tr>
<tr>
<td>#501972</td>
<td>NEW</td>
<td>12013</td>
<td>7765</td>
<td>350</td>
<td>6197 L: 26 j6AFmerim/cAesica/DoNj6 Ge</td>
<td></td>
</tr>
<tr>
<td>#506861</td>
<td>NEW</td>
<td>12013</td>
<td>7770</td>
<td>351</td>
<td>6257</td>
<td>32 19 [-6'7.041 06::9 6::.8AFmqF7</td>
</tr>
<tr>
<td>#507341</td>
<td>NEW</td>
<td>12013</td>
<td>7771</td>
<td>352</td>
<td>6263 L: 31 6,F:6 ,6F ,!F:6 ,F:-9.N:-9..:N</td>
<td></td>
</tr>
<tr>
<td>#509461</td>
<td>NEW</td>
<td>12013</td>
<td>7772</td>
<td>353</td>
<td>6212 L: 30 197.041 q7:3_+10:9!CLSTYST2F:0</td>
<td></td>
</tr>
<tr>
<td>#517081</td>
<td>NEW</td>
<td>12013</td>
<td>7774</td>
<td>354</td>
<td>6229 L: 31 am'-FGFF8GFYJF -HH24-.:text-FYF</td>
<td></td>
</tr>
<tr>
<td>#524288</td>
<td>pulse</td>
<td>12013</td>
<td>7774</td>
<td>354</td>
<td>6241</td>
<td></td>
</tr>
<tr>
<td>#556354</td>
<td>NEW</td>
<td>12013</td>
<td>7775</td>
<td>355</td>
<td>6251 L: 27 201-#0-3720 02*01-0;(&amp;(d9-</td>
<td></td>
</tr>
<tr>
<td>#574472</td>
<td>NEW</td>
<td>12013</td>
<td>7776</td>
<td>356</td>
<td>6244 L: 27 9:MY7(7 ,^:;(;'.+Tt7*.t+J.M</td>
<td></td>
</tr>
<tr>
<td>#580776</td>
<td>NEW</td>
<td>12013</td>
<td>7777</td>
<td>357</td>
<td>6244 L: 26 2011-020-071-00902011-0199</td>
<td></td>
</tr>
</tbody>
</table>
CREATE FUNCTION fuzz()
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

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Condition Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>22007</td>
<td>invalid_datetime_format</td>
</tr>
<tr>
<td>22008</td>
<td>datetime_field_overflow</td>
</tr>
<tr>
<td>22009</td>
<td>invalid_time_zone_displacement_value</td>
</tr>
<tr>
<td>0A000</td>
<td>feature_not_supported</td>
</tr>
<tr>
<td>22023</td>
<td>invalid_parameter_value</td>
</tr>
</tbody>
</table>
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;

Found by LLVM's Libfuzzer
Fix regular-expression compiler to handle loops of constraint arcs.

Add recursion depth protections to regular expression matching.

Fix potential infinite loop in regular expression execution.

Add some more query-cancel checks to regular expression matching.

Docs: add disclaimer about hazards of using regexps from untrusted sources.
commit 9fe8fe9c9a5d7fc099acfc96e976ee72b2b49865
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 13:45:39 2015 -0400

Add some more query-cancel checks to regular expression matching.

commit 558d4ada1851274fe4dd3618f3f6561b63857e8f
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Oct 2 13:30:42 2015 -0400

Docs: add disclaimer about hazards of using regexps from untrusted sources.
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