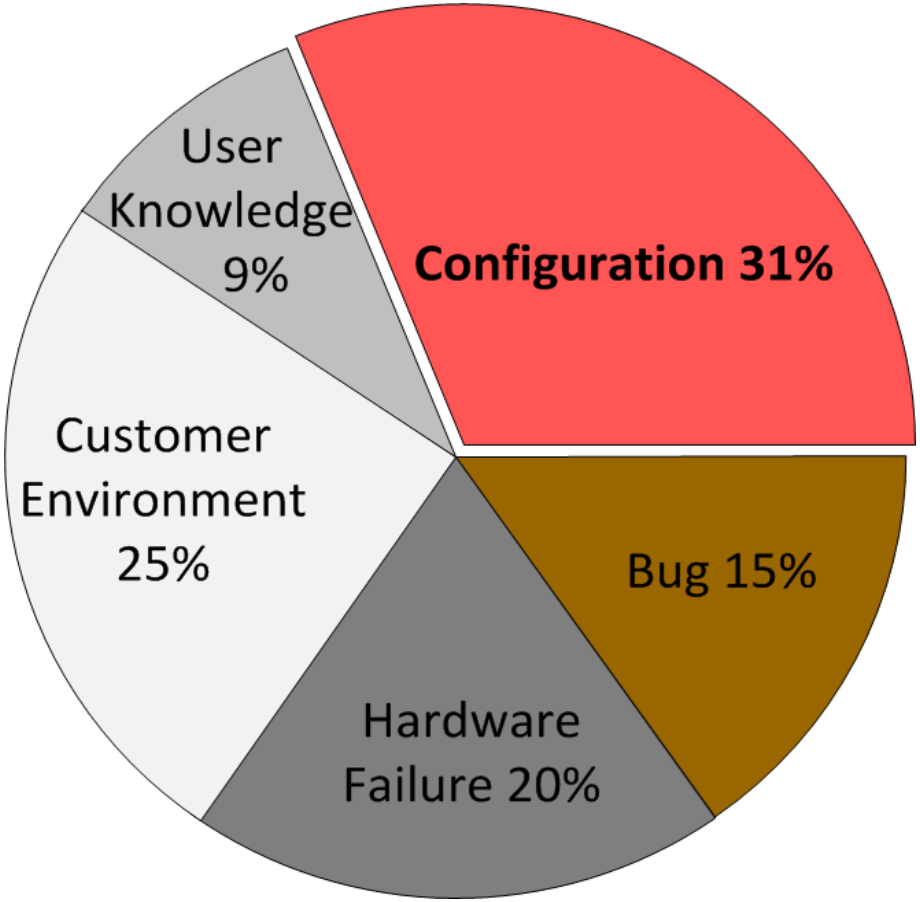
Do Not Blame Users for Misconfigurations

Tianyin Xu, Jiaqi Zhang, Ryan Huang
Jing Zheng, Tianwei Sheng, Ding Yuan,
Yuanyuan Zhou, Shankar Pasupathy*

UC San Diego **NetApp*

How many of you have made mistakes when configuring systems?

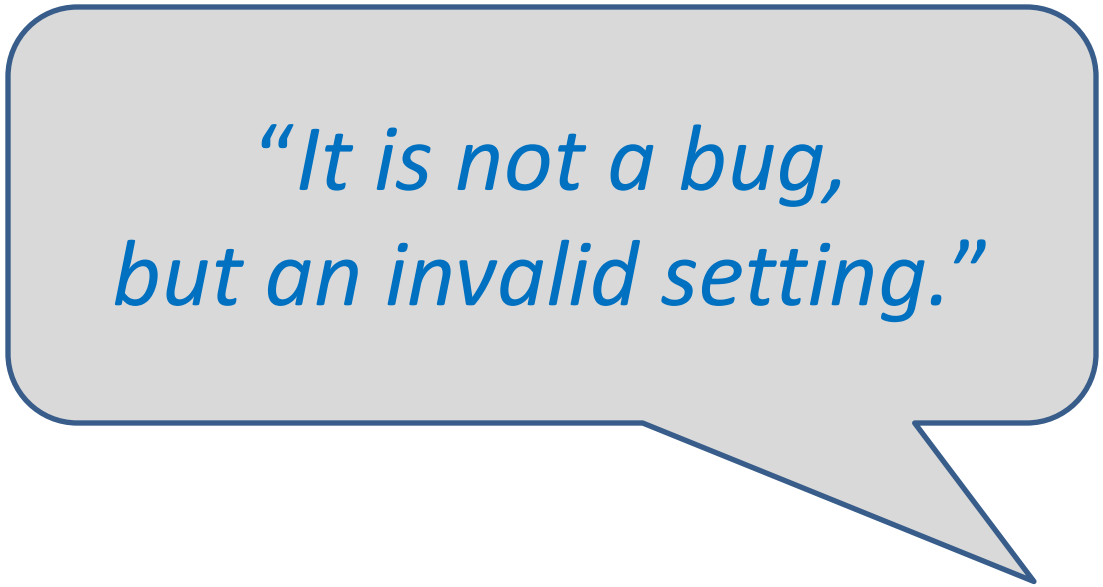
You Are Not Alone!



Root causes of "high-severity" customer issues in a major storage company [Yin et al, SOSP'11]

How many of you think your misconfig. was your fault?

Unfortunately, many developers think they are users' faults!



Developers of a mature open-source server app.

Configuration Is a User Interface!

```
datadir    = /var/lib/mysql
tmpdir     = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address    = 127.0.0.1
#
# * Fine Tuning
#
key_buffer    = 16M
max_allowed_packet  = 16M
thread_stack    = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover         = BACKUP
#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit = 1M
query_cache_size        = 16M
#
# * Logging and Replication
```

Goal #1: React Gracefully to Misconfig.

- Today's systems are **vulnerable** to misconfig.

Software Systems	Crashes & Hangs w/o Message
Storage-A	8.4%
CentOS	6.7%
MySQL	16.4%
Apache	5.0%
OpenLDAP	4.8%

The impact distribution of misconfigurations [Yin et al, SOSP'11]

Goal #2: Intuitive & Less Error-prone

/* A Commercial Storage System*/

InitiatorName = iqn_DEV_domain

Error!

Lower-case only

Error-prone constraint

Several customers made the same mistakes.

Our Contributions

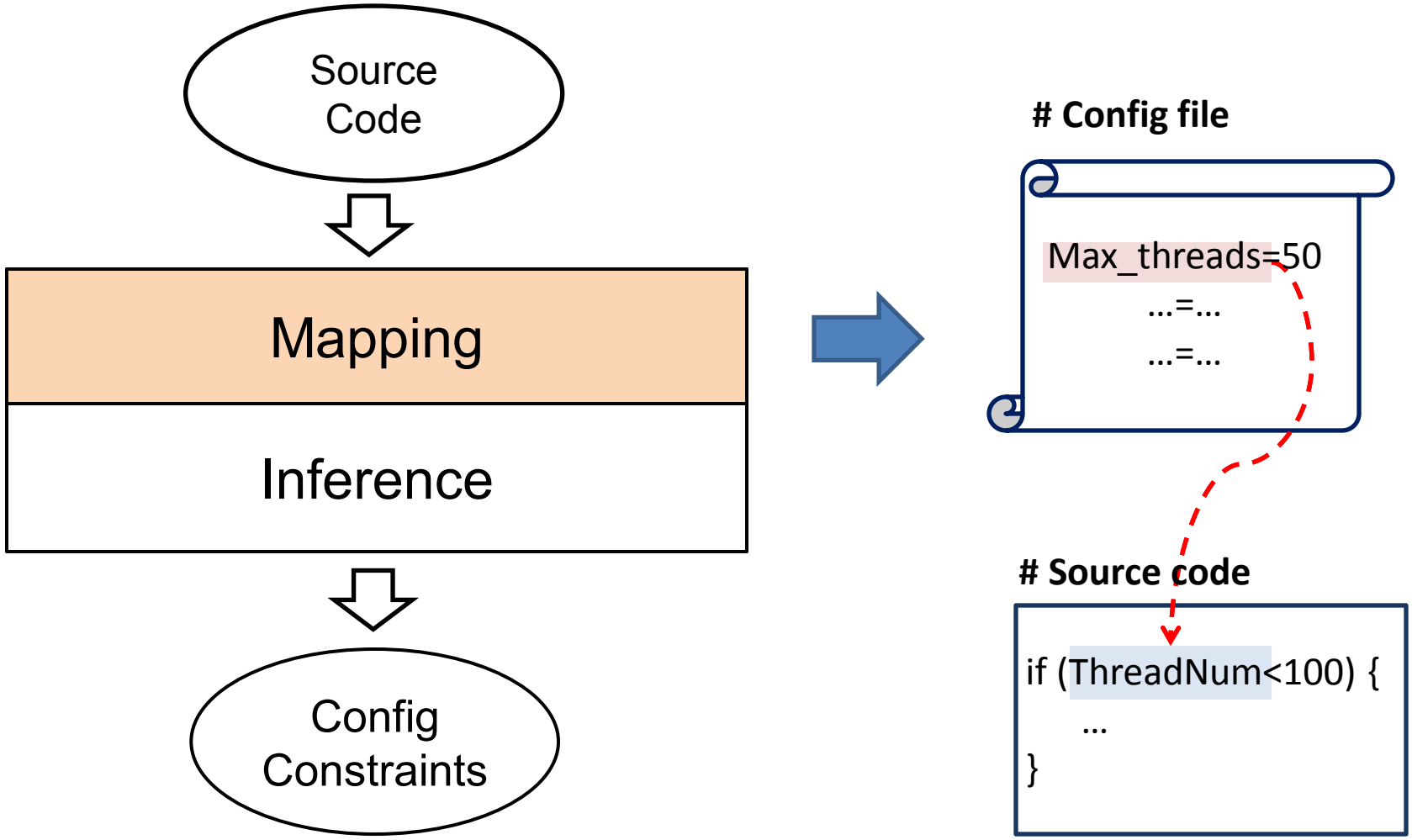
1. ***Spex***: automatically infer config. constraints
   by statically analyzing source code (for developers)

2. Use cases
   - Expose misconfig. vulnerabilities
   - Detect error-prone config. design & handling

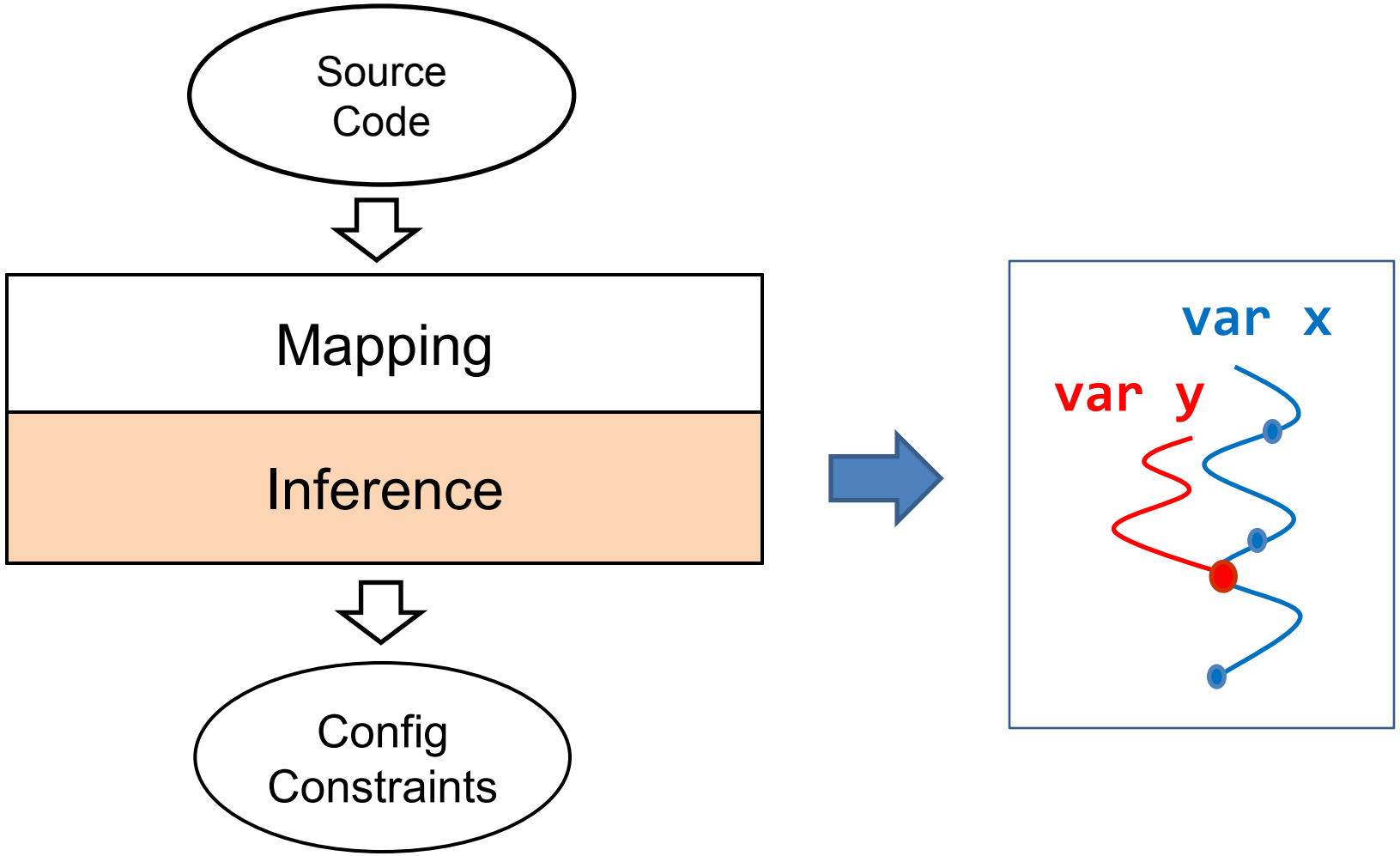
3. Improve config. design of real-world systems
   - 1 commercial and 6 open-source systems
   - Expose 743 vulnerabilities (364 confirmed/fixed)
   - Detect 112 error-prone constraints (80 fixed)

4. Experience in interacting with developers
   - Improve Squid's config. lib (benefit 150+ parameters)

Spex Overview



Spex Overview



Mapping Is Non-trivial

- Cannot ask developers to annotate every parameter
- Investigated 18 software projects, all but one use one of the following mapping conventions.

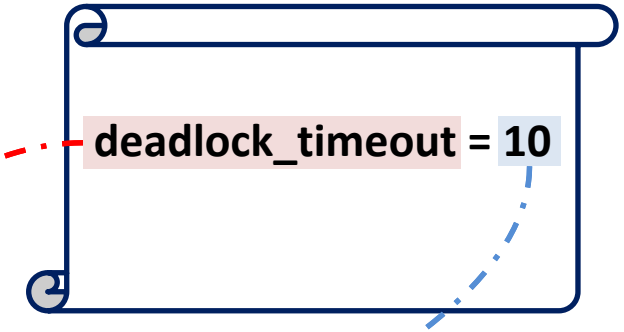
Mapping convention	What need to be annotated?	# Software projects
Structure-based	Data structure(s)	9
Comparison-based	Parsing function	4
Container-based	Getter functions	4

Structure-based Mapping

- **PostgreSQL-9.2.1**

```
struct config_int
ConfigureNamesInt[] =
{
    {“deadlock_timeout”,
     …,
    &DeadlockTimeout, …, },
   …
   {“max_connections”,
     …,
    &MaxConnections, …, },
   …
 }
```

80 more mappings



Annotation

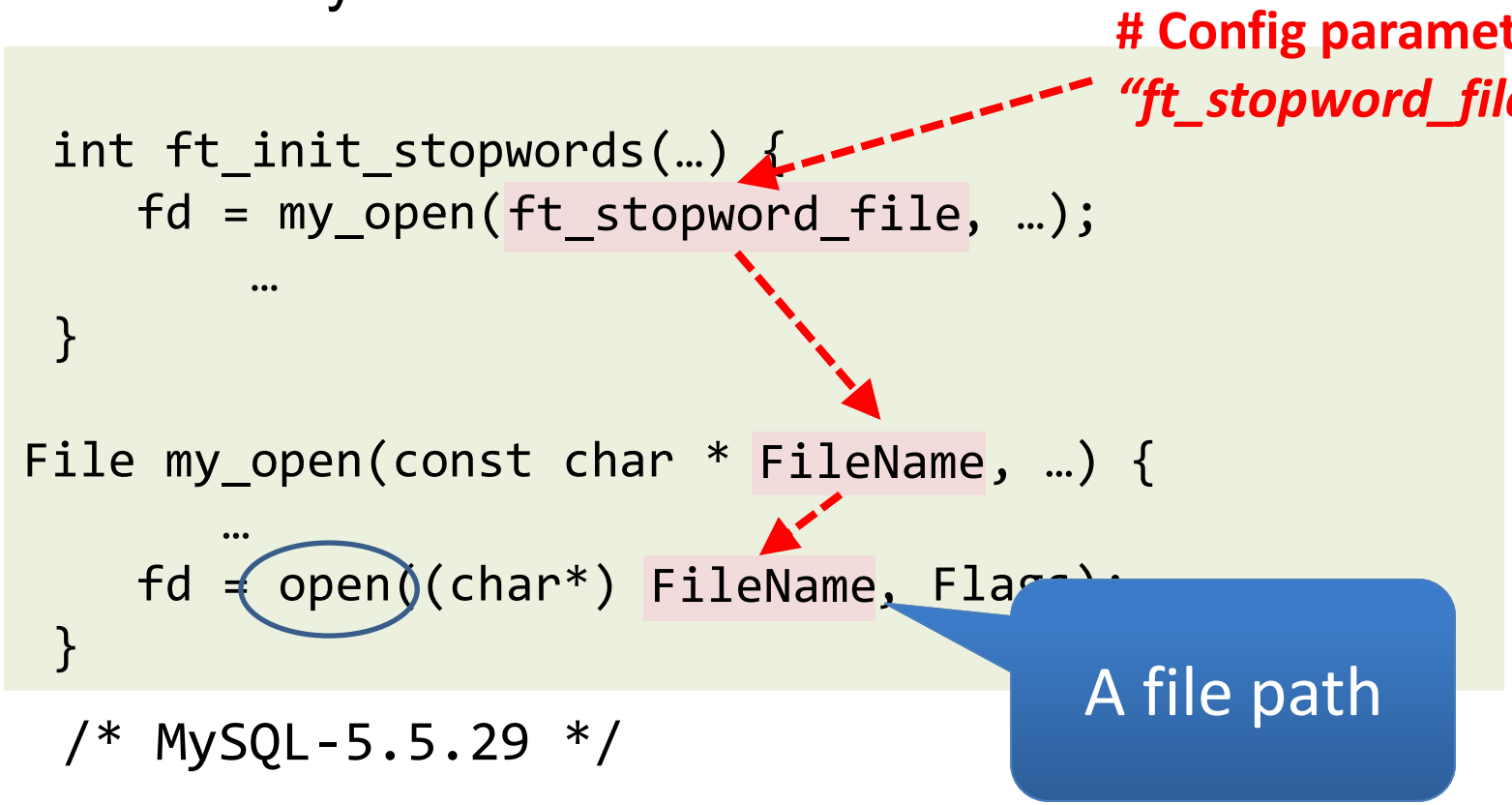
@STRUCT = ConfigureNamesInt
@PAR = [config_int, 1]
@VAR = [config_int, 3]

What Constraints Can Be Inferred?

1. **Data type**
   - *e.g., integer, float, string, boolean*
     file path, IP address, port
2. **Data range**
   - *e.g., [10, 100], {'yes', 'no'}*
3. **Control dependency**
   - *e.g., X dominates Y's executions*
4. **Value relationship**
   - *e.g., X < Y*

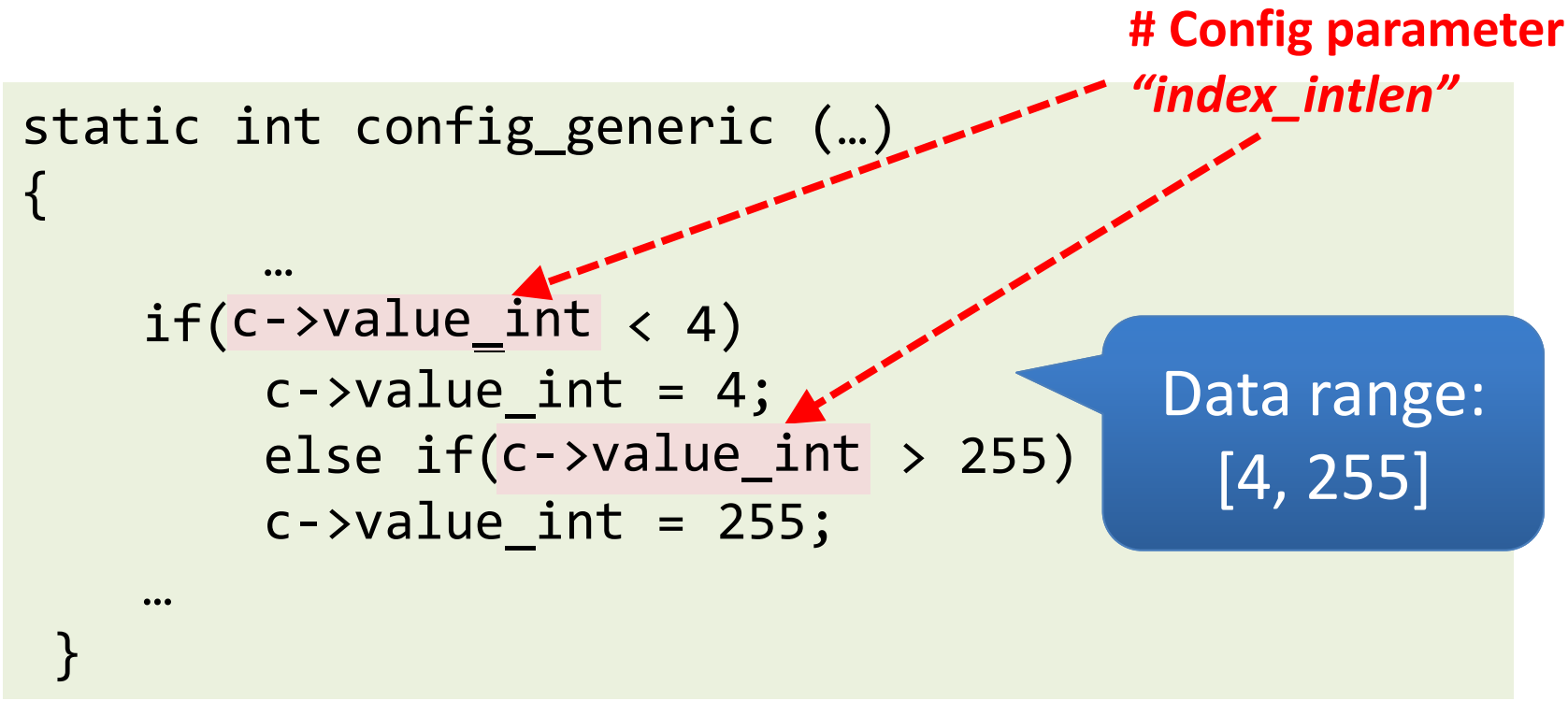
Data Type Inference

- Methodology
  - Check the variable's data type and how the variable is used in syscall/libcall



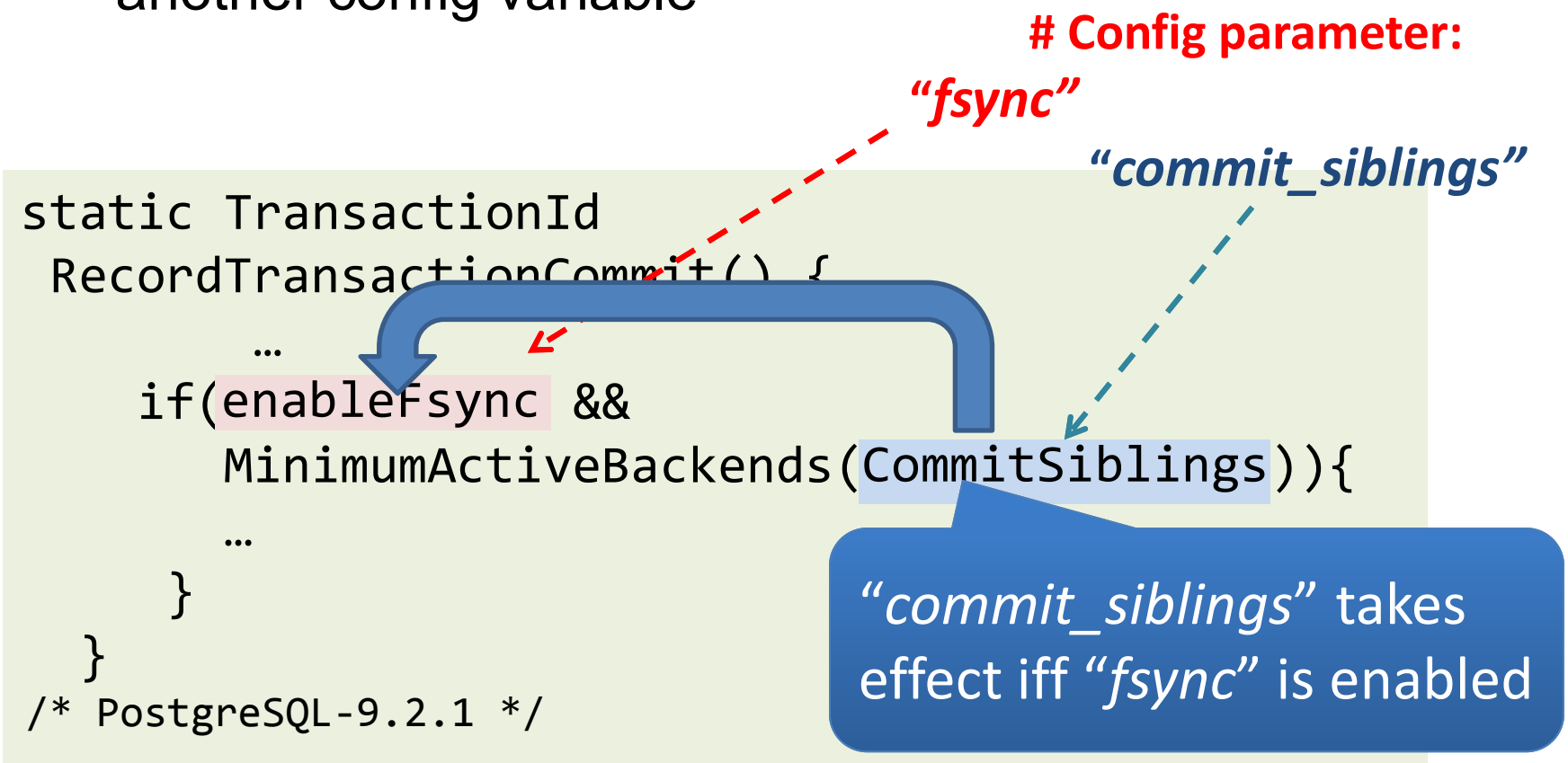
Data Range Inference

- Methodology
  - If the variable is compared with a constant value, inspect the branch block to decide the range.



Control Dependency Inference

- Methodology
  - Check if the config variable's usage is controlled by another config variable



*All commit_siblings's use sites are inside the func call.

Use Case of Constraints #1

- Expose misconfig. vulnerabilities
  - Misconfig. injection testing

Type	Constraint	Config. Error
Basic type	A is an integer	A := 2XX
Semantic type	B is a file path	B := invalid path
Data range	C ∈ [10, 100]	C := 1000
Ctrl dep.	D depends on E	D := yes, E := no
Value rel.	F > H	F < H

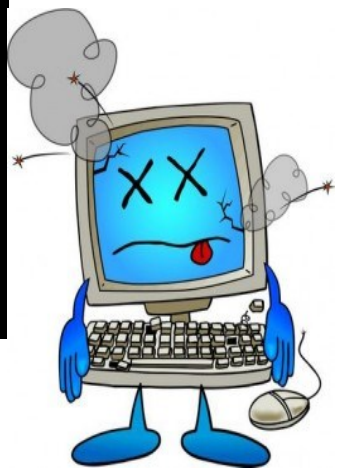
Implemented as a plugin framework (easy to extend)

Expose Vulnerability

MySQL-5.5.29

```
ft_stopword_file dir_path
```

```
Program received signal SIGSEGV,
Segmentation fault.
my_mb_ctype_8bit (cs=0x1226760,
ctype=0x7ffffffde00, s=0x1ad5000 <Address
0x1ad5000 out of bounds>, e=0x10185a67f
<Address 0x10185a67f out of bounds>) at
./strings/ctype-simple.c:1299 1299 *ctype=
cs->ctype[*s + 1];
……
```



Use Case of Constraints #2

- Detect error-prone config. design & handling
  - Design inconsistency (case sensitivity, unit)
  - Silent overruling
  - Undocumented constraints

Software Evaluated

Software	Proprietary	LOC	# Parameters	LOA*
Storage-A	Commercial	--	> 1000	5
Apache	Open source	148K	103	4
MySQL	Open source	1.2M	272	29
PostgreSQL	Open source	757K	231	7
OpenLDAP	Open source	292K	86	4
VSFTP	Open source	16K	124	5
Squid	Open source	180K	335	2

*LOA: lines of annotation

Exposed Misconfig. Vulnerabilities

Software	Crash/ Hang	Early Termina.	Function Failure	Silent Violation	Silent Ignor.	Total
Storage-A	0	0	7	74	83	**164**
Apache	5	4	9	29	5	**52**
MySQL	5	10	12	71	16	**114**
PostgreSQL	1	10	2	1	35	**49**
OpenLDAP	1	3	6	7	0	**17**
VSFTPD	12	5	18	23	68	**126**
Squid	2	3	29	173	14	**221**
Total	26	35	83	378	221	**743**

Detected Inconsistency

Software	Case Sensitivity		Fixed parameters
	Sensitive	Insensitive	
Storage-A	32 (7.1%)	453 (92.3%)	0
Apache	3 (11.5%)	26 (88.5%)	3
MySQL	1 (1.7%)	58 (98.3%)	1
PostgreSQL	0 (0%)	92 (100%)	N/A
OpenLADP	0 (0%)	9 (100%)	N/A
VSFTP	0 (0%)	73 (100%)	N/A
Squid	85 (52.8%)	76 (47.2%)	76
Total			80

Can We Help Real-world Misconfig.?

Software	Real-world misconfig.	Bad reactions that can be potentially avoided
Storage-A	246	68 (27.6%)
Apache	50	19 (38.0%)
MySQL	47	14 (29.8%)
OpenLDAP	49	12 (24.5%)

Inference Accuracy

Software	Basic Type	Semantic Type	Data Range	Control Dep.	Value Dep.
Storage-A	97.0%	95.7%	87.1%	84.1%	94.7%
Apache	96.1%			100.0%	81.8%
MySQL	100.0%			94.7%	71.4%
PostgreSQL	100.0%			91.7%	85.7%
OpenLDAP	88.2%	93.7%	73.1%	N/A	50.0%
VSFTP	100.0%	100.0%	100.0%	63.9%	100.0%
Squid	77.0%	100.0%	100.0%	77.8%	100.0%

Avg. 90.6%

Experience (Positive ☺)

- **Storage-A:**
  - Slides sent to all the developers
- **Squid:**
  - Improve the config. lib. (150 parameters benefit)
- 364 detected misconfig. vulnerabilities have been confirmed or fixed by developers.
- 80 detected error-prone constraints have been fixed by developers.

Experience (Negative ☹)

"It is not a bug, but an invalid setting."

"Those who do (configuration) typically read the code."

"If you work exactly and carefully it does not matter; if not, you should not maintain a server at all."

Limitations

1. We cannot infer all the constraints, e.g., domain-specific, cross-software
2. The inference is not 100% accurate
3. More fundamental approach is to rethink and redesign of configuration

Related Work

- Detection and Diagnosis
  - Detection: [Feamster NSDI'05], [Yuan USENIX'11],
  - Diagnosis: [Wang OSDI'04], [Witaker OSDI'04], [Attariyan OSDI'10], [Attariyan OSDI'12]
- Testing system resilience to config errors
  - Mutation testing: [Keller DSN'08]
- Extract source code information for config
  - Type information: [Rabkin ICSE'11]

Conclusions

- Take a more active role in handling misconfig.
  - Configuration is a user interface!
- Spex: a tool that automatically infers config. constraints from source code.
  - Exposed 741 vulnerabilities (364 confirmed/fixed)
  - Detected 112 error-prone constraints (80 fixed)