# Securing Web Applications with Predicate Access Control

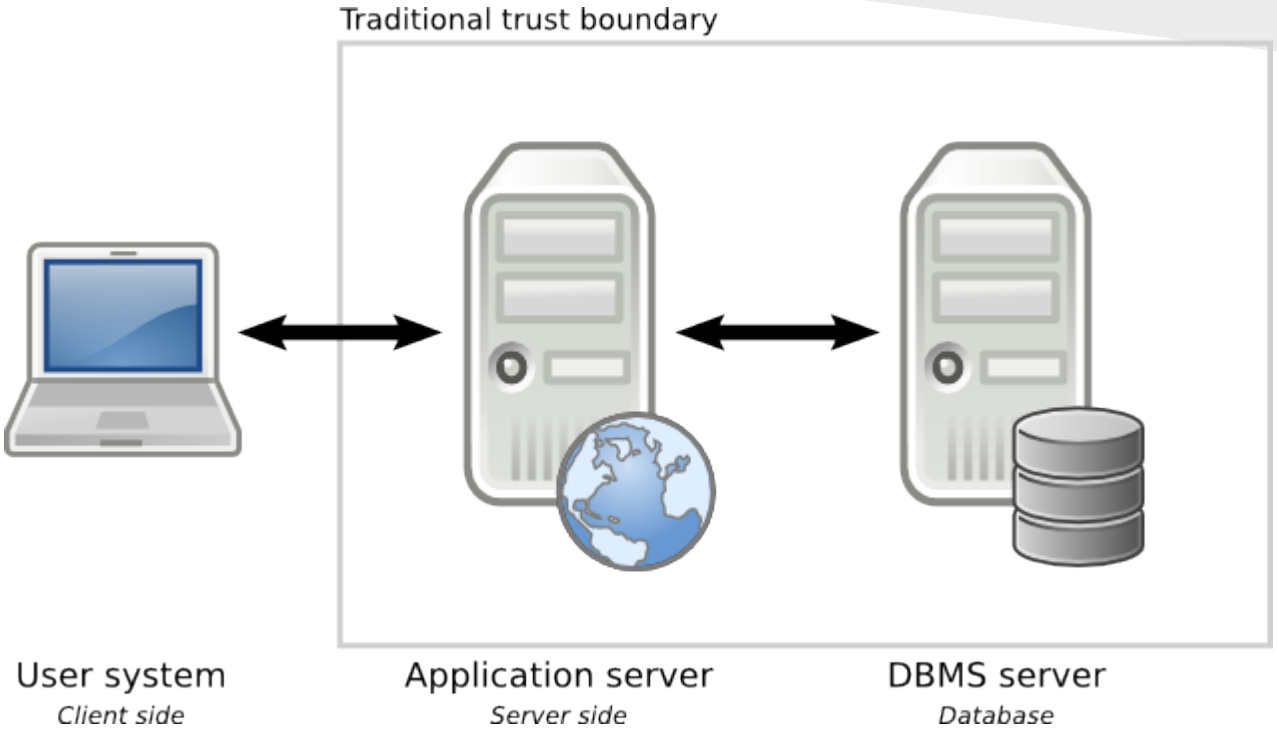
*Zhaomo Yang  Kirill Levchenko*

UCSDCSE
Computer Science and Engineering

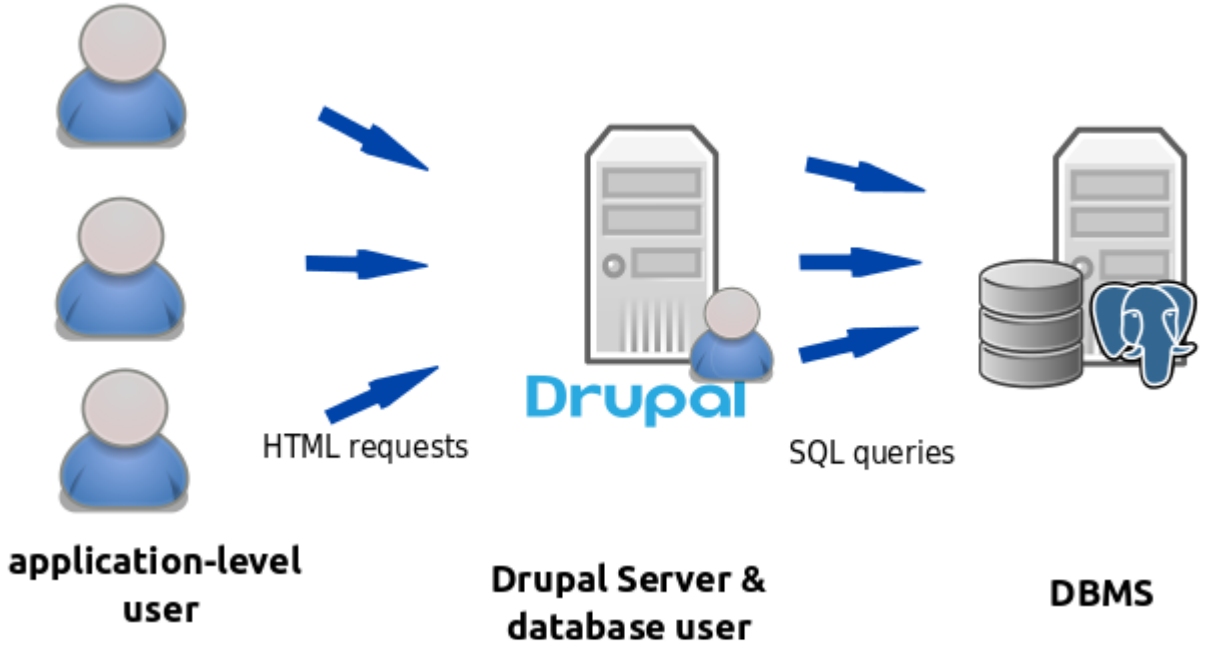
# Observation 1: Vulnerabilities

Web applications are broken. There are so many vulnerabilities!

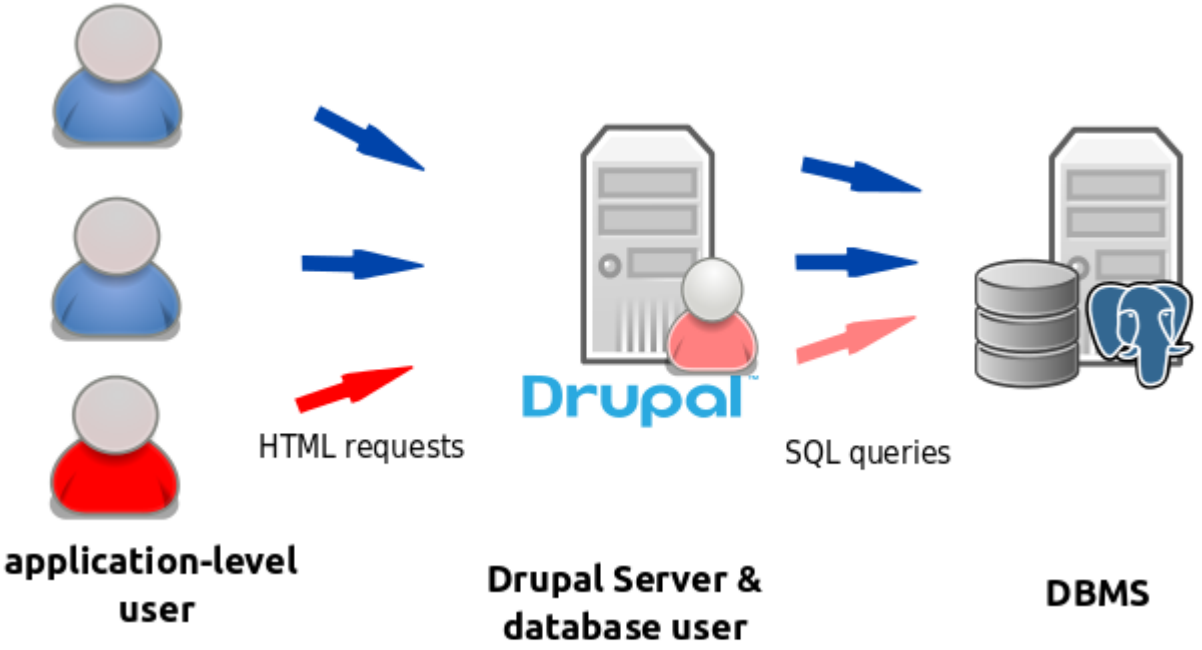
# Observation 2: Access Control



# Observation 2: Access Control



# Observation 2: Access Control



# Previous Work

- Nemesis [4]
- GuardRails [5]
- CLAMP [6]
- Authorization views [1]
- Parametrized views with authentication token [2]
- Oracle's Virtual Private Database

…

# EBay asks 145 million users to change passwords after cyber attack

BY **JIM FINKLE**, **SOHAM CHATTERJEE** AND LEHAR MAAN
**BOSTON/BANGALORE** Wed May 21, 2014 4:25pm EDT

# App Behind The Snapchat Leak Admits It Was Hacked, Apologizes

Posted: 10/13/2014 1:32 pm EDT | Updated: 10/13/2014 1:32 pm EDT

# GitHub hacked, millions of projects at risk of being modified or deleted

By Sebastian Anthony on March 5, 2012 at 7:22 am | 22 Comments

# Magento Go vulnerability allows hackers to bypass the authentication by creating administrator account

Posted by snoopy On February 15, 2014 In Vulnerability No comments

# 50,000 sites hacked through WordPress plug-in vulnerability


Lucian Constantin
Jul 24, 2014 10:38 AM

# Question?

What hinders the adoption of security mechanisms for web applications?

# Hypothesis

To be widely adopted, security mechanisms must be first and foremost easy to understand and use for developers.

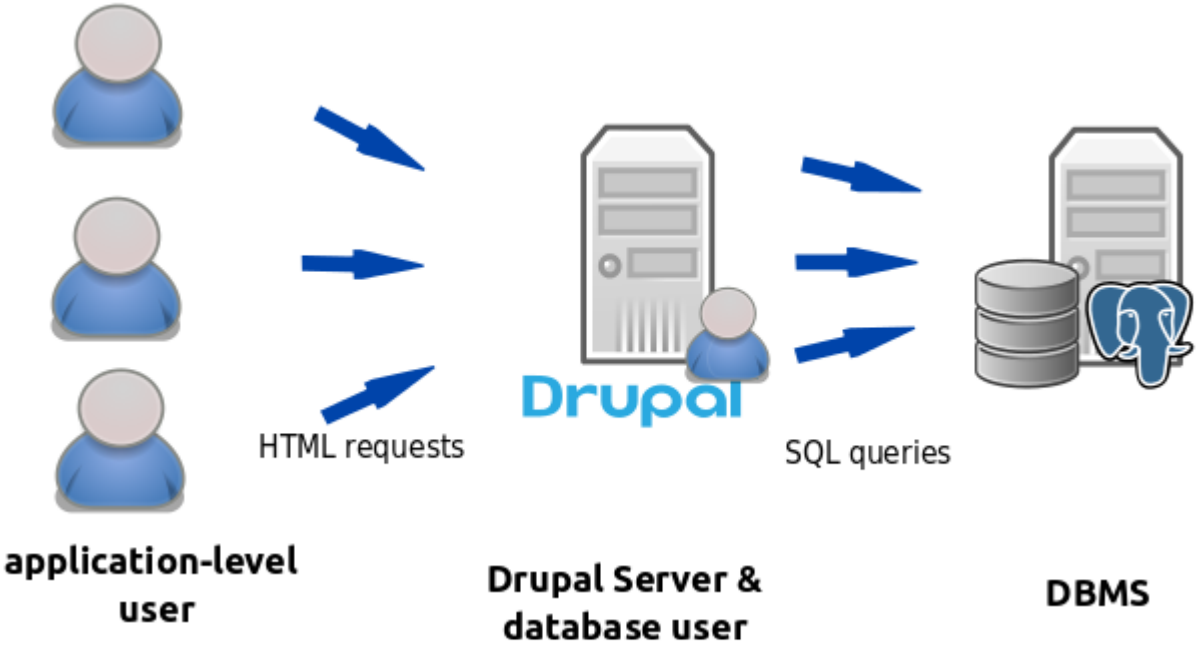
# The plan

- Build it
  - we built a security mechanism with usability in mind.
- Field it
  - make it available to open source web app communities
- Evaluate
  - measure adoption and efficacy

# Build it

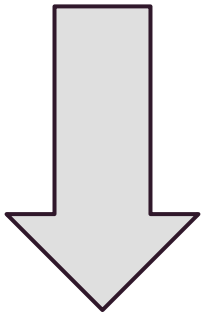
Build a mechanism **in DBMS** which protects app users from each other.

# Structure of Web Apps



# Build it - Authentication

Problem: DBMS has no notion of application users.



We need to authenticate application users to DBMS

# Toy Example

GradeBook

- Student: view his/her grades
- Teacher: view/post/update/ students' grades

# Authentication in Web Apps

In web apps, authentication usually involves

- Get user's id and password;
- Query table to see if it is a registered application user;
  - We call it the "**authentication query**".
- If it is, save the identity; If not, deny login.

# Example of Authentication Query

```
SELECT user_id, instr FROM Users
WHERE user_name = $1 AND
        pass_hash = SHA2(pass_salt || $2);
```

# Build it - Authentication

Now our problem becomes: how to let the DBMS know the current application user's identity.

# Build it - Authentication Function

## CREATE AUTHENTICATION FUNCTION

Requirement:

- Wrap the authentication query which returns the identity of current application user

# Example of Authentication function

```
CREATE AUTHENTICATION FUNCTION Auth(TEXT, TEXT)
RETURNS TABLE (user_id INTEGER, instr BOOLEAN)
AS $$
  SELECT user_id, instr FROM Users
  WHERE user_name = $1 AND
                pass_hash = SHA2(pass_salt || $2);
$$ LANGUAGE SQL;
```

# Build it - Access Control

- Traditional SQL GRANT statement

```
GRANT [privilege] ON [table] TO [database user];
```

E.g.

```
GRANT SELECT ON grades TO gradebook;
```

# Build it - Access Control

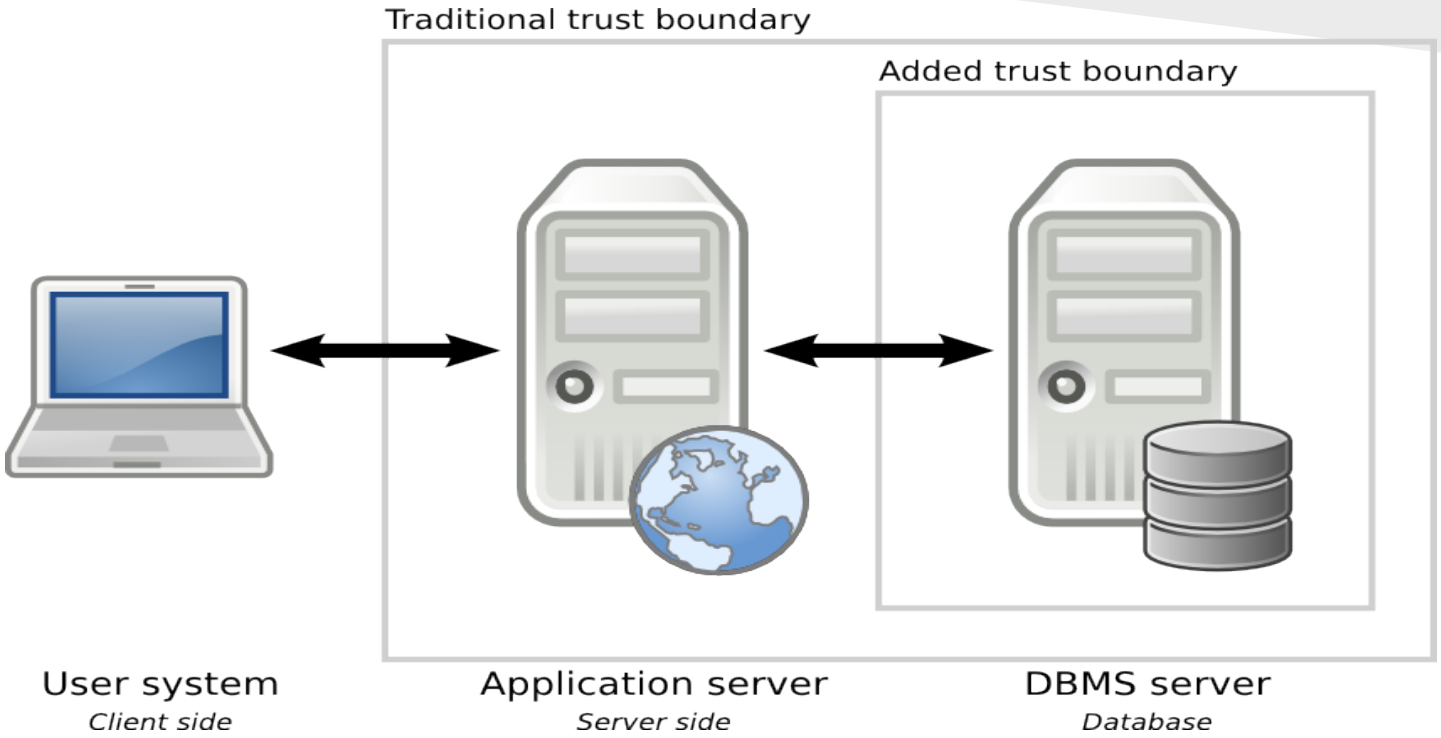
- GRANT-WHERE statement proposed by [3]

```
GRANT [privilege] ON [table] TO [database user]
USING [referenced tables ... ]
WHERE [row-level predicate];
```

E.g.

```
GRANT SELECT ON grades TO gradebook
USING Auth
WHERE Auth.instr = false AND
        Auth.user_id = grades.user_id;
```

# Our Mechanism - Trust Boundary



# Implementation

We implemented the mechanism as an extension to PostgreSQL 9.2

- Temporary Rules
- Compiler

# Implementation - Temporary Rules

- Rule System in PostgreSQL

In PostgreSQL, every query is rewritten against the rules defined on the same table

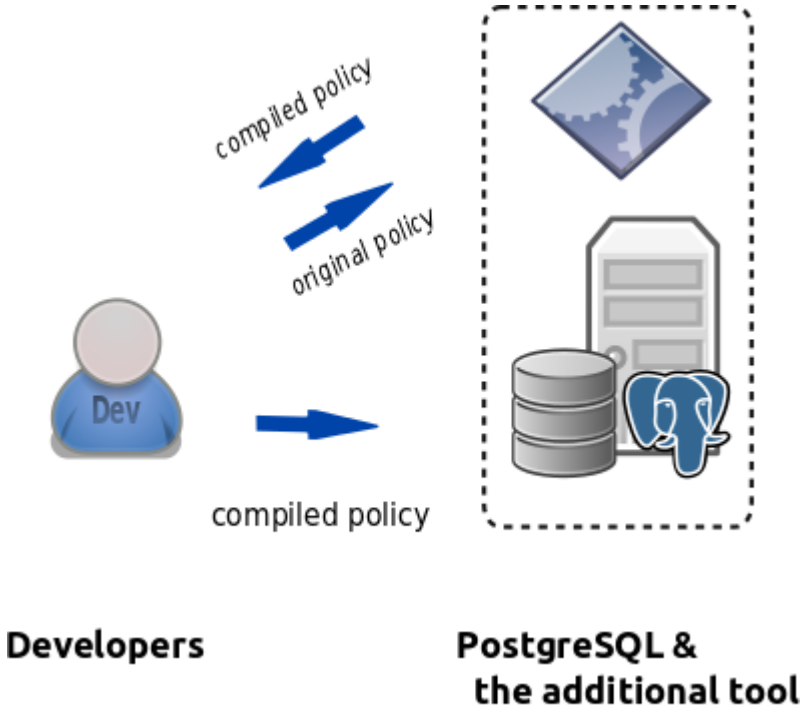
# Implementation - Translation

- How a GRANT-WHERE statement is translated

```
GRANT UPDATE ON Grades TO Gradebook
USING Auth
WHERE Auth.instr;
```

```
CREATE RULE _UPDATE AS ON UPDATE TO Grades
WHERE NOT EXISTS (SELECT 1 FROM Auth
                  WHERE current_user = Gradebook AND Auth.instr)
DO INSTEAD NOTHING;
```

# Implementation - Compiler



# Evaluation

- Criteria: Is it expressive enough?
- Used our mechanism to protect Drupal 7.1 and Spree e-commerce

# Evaluation

```
GRANT SELECT ON spree_payments TO mystore
USING Auth, spree_orders
WHERE Auth.is_admin = true
    OR (spree_orders.user_id = Auth.user_id
        AND spree_payments.order_id = spree_orders.id);
```

# Evaluation - Performance

| Task | Without policy | With policy |
|---|---|---|
| Drupal view article | 0.44 s | 0.54 s |
| Drupal edit article | 1.29 s | 1.61 s |
| Spree buy item | 18.40 s | 24.12 s |

# Evaluation - Result

| App. Name | CVE # | Description |
|---|---|---|
| Drupal | CVE-2012-1590 | User permissions are not checked properly for unpublished forum nodes, which allows remote authenticated users to obtain sensitive information such as the post title via the forum overview page. |
| Drupal | CVE-2012-2153 | node_access is not added to queries thus queries are not rewritten properly and restrictions of content access module are ignored. |
| Drupal | CVE-2011-2687 | Proper tables are not joined when node access queries are rewritten thus access restrictions of content access module are ignored. |
| Spree | No CVE ID | By passing a crafted string to the API as the API token, a user may authenticate as a random user, potentially an administrator. |
| Spree | CVE-2013-2506 | Mass assignment is not performed safely, which allows remote authenticated users to assign any roles to themselves. |

# The plan

- Build it
  - we built a security mechanism with usability in mind.
- Field it
  - make it available to open source web app communities
- Evaluate
  - measure adoption and efficacy

# Thanks!

# Reference

[1] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending Query Rewriting Techniques for Fine-Grained Access Control. In

Proceedings of the 2004 ACM SIGMOD international conference on Management of Data, pages 551–562, June 2004.

[2] A. Roichman and E. Gudes. Fine-grained Access Control to Web Databases. In Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT), pages 31–40, June 2007.

[3] S. Chaudhuri, T. Dutta, and S. Sudarashan. Fine Grained Authorization Through Prediidation Vulnerabilities in Web Applications.

cated Grants. In Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE), pages 1174–1183, April 2007.

[4] M. Dalton, C. Kozyrakis, and N. Zeldovich. Nemesis: Preventing Authentication & Access Control Vulnerabilities in Web Applications. In Proceedings of the 18th USENIX Security Symposium, August 2009.

[5] J. Burket, P. Mutchler, M. Weaver, M. Zaveri, and D. Evans. GuardRails: A Data-Centric Web Application Security Framework. In Proceedings of the 2nd USENIX Conference on Web Application Development (WebApps), June 2011.

[6] B. Parno, J. McCune, D. Wendlandt, D. Andersen, and A. Perrig. CLAMP: Practical Prevention of Large-Scale Data Leaks. In Proceedings of the 30th IEEE Symposium on Security & Privacy, pages 154–169, 2009.

[7] J. Tudor, Web ApplicationVulnerability Statistics 2013, Context Information Security, June 2013.