Synthesis of Allowlists for Runtime Protection Against SQLi

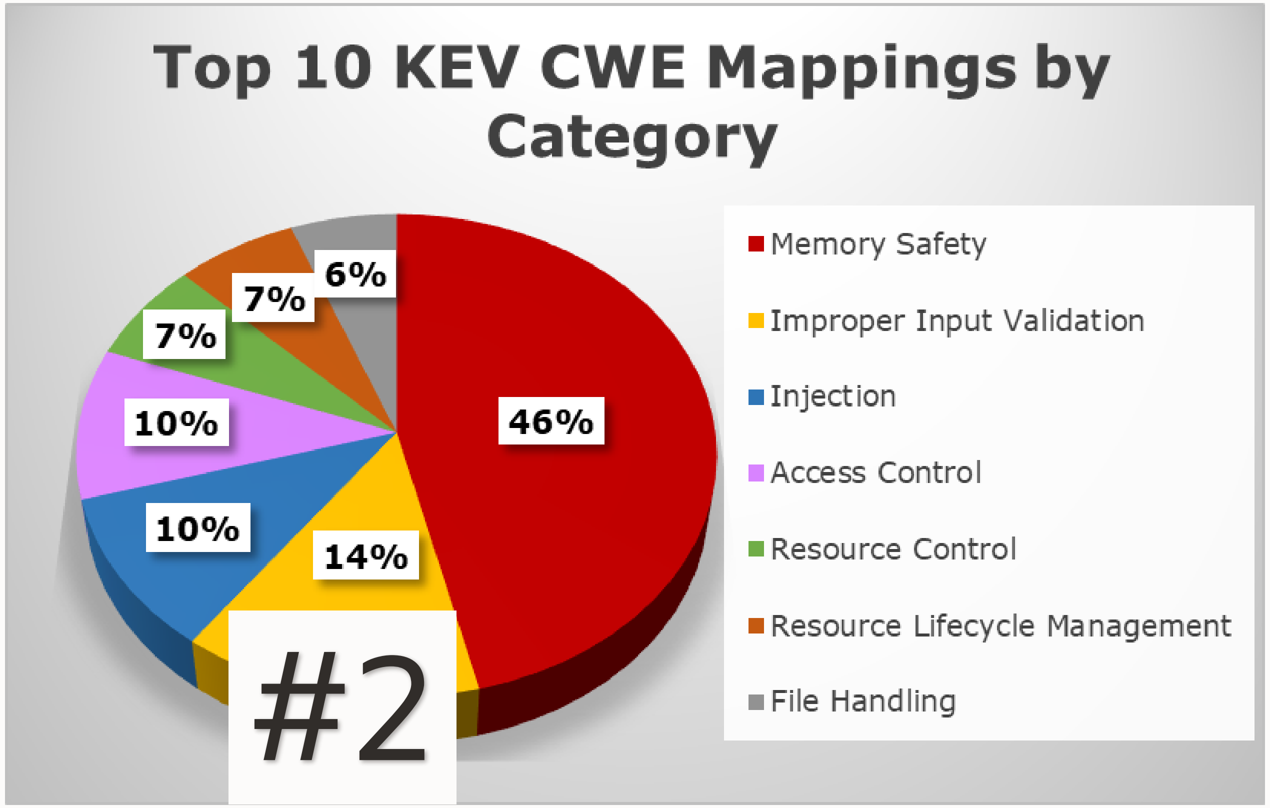
ICSE NIER '24

Kostyantyn Vorobyov, **François Gauthier,** and Padmanabhan Krishnan
Senior Principal Researcher
Oracle Labs Australia

SQL Injection is Still a Thing… Unfortunately

#3

#3



Known Exploited Vulnerabilities (KEV) catalog

State-of-the-Art SQLi Detection

Lexical, syntactic and feature-based approaches

```
SELECT * FROM Users WHERE uname=‘John’ AND passwd=‘1234’ OR 1=1; --’;
```

Lexical

(e.g. modsecurity[1])

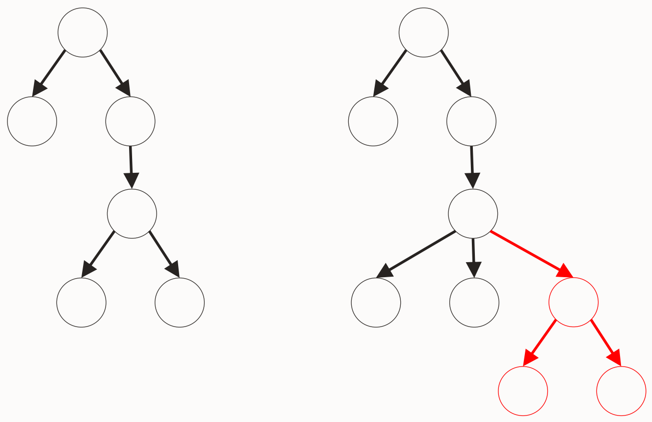
Rule id 981231:
[Detect SQL Comment Sequence]

```
(/\*!?|\*/|[';]--|--
[\s\r\n\v\f]|(?:--[^-]*?-)|([^\-
&])#.*?[\s\r\n\v\f]|;?\\x00)
```

Syntactic

(e.g. Sofia[2], SEPTIC[3])

Detect parse tree anomalies:



Feature-based

(e.g. SQLBlock[4])

Detect anomalous SQL features:

Query type	SELECT
Selected	*
Table name	Users
Conditional	AND, **OR**
Equality	FIELD=LITERAL

Proxy Measures Indirectly Capture Information Disclosure

The SQL injection contrast:

- SQLi *execution* requires syntactic modification of a legitimate query.
- SQLi *consequence* is unwanted information disclosure, insertion and removal, or remote code execution.

State-of-the-art makes two fundamental assumptions:

1. Unwanted information disclosure is only achieved through anomalous syntactic features.
2. Legitimate information disclosure is only achieved through “normal” syntactic features.

Complex applications can break both assumptions:

1. Seemingly benign syntactic changes can lead to serious consequences (false negative).
2. Rare and re-factored legitimate queries can appear anomalous (false positive).

Let’s consider a simple example.

Inferring Acceptable Disclosure From Training Queries (1)

- **Training query 1:**
```
SELECT lname, fname, email FROM members
    WHERE dept='HR';
```

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Inferring Acceptable Disclosure From Training Queries (2)

- Training query 1:
```
SELECT lname, fname, email FROM members
    WHERE dept='HR';
```

- **Training query 2:**
```
SELECT dept, email FROM members WHERE
    lname='Baker' OR dept='Sales';
```

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Inferring Acceptable Disclosure From Training Queries (3)

Cells from partially disclosed rows and columns are inferred acceptable

- Training query 1:
```
SELECT lname, fname, email FROM members
    WHERE dept='HR';
```

- Training query 2:
```
SELECT dept, email FROM members WHERE
    lname='Baker' OR dept='Sales';
```

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Inferring Acceptable Disclosure From Training Queries (3)

Blue cells can be disclosed, red cells are confidential

- Training query 1:
```
SELECT lname, fname, email FROM members
    WHERE dept='HR';
```
- Training query 2:
```
SELECT dept, email FROM members WHERE
    lname='Baker' OR dept='Sales';
```

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Syntactic Features are no Substitute for Information Disclosure

False negative from feature-based approaches

- Incoming query 1:
  `SELECT email FROM members;`
  - Fewer features, more information disclosed

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Syntactic Features are no Substitute for Information Disclosure

False positive from feature-based approaches

- Incoming query 1:
  `SELECT email FROM members;`
  - Fewer features, more information disclosed

- Incoming query 2:
  `SELECT email FROM members WHERE dept=‘HR’ AND lname=‘Baker’;`
  - More features (AND), less information disclosed

fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Syntactic Features are no Substitute for Information Disclosure

False positive from syntax-based approaches

- Incoming query 1:
  `SELECT email FROM members;`
  - Fewer features, more information disclosed
- Incoming query 2:
  ```
  SELECT email FROM members WHERE dept='HR'
      AND lname='Baker';
  ```
  - More features (AND), less information disclosed
- Incoming query 3:
  ```
  SELECT fname || ' ' || lname AS name
      FROM Members M HAVING dept
      IN ('Sales', 'HR');
  ```
  - Different syntax, same information disclosed

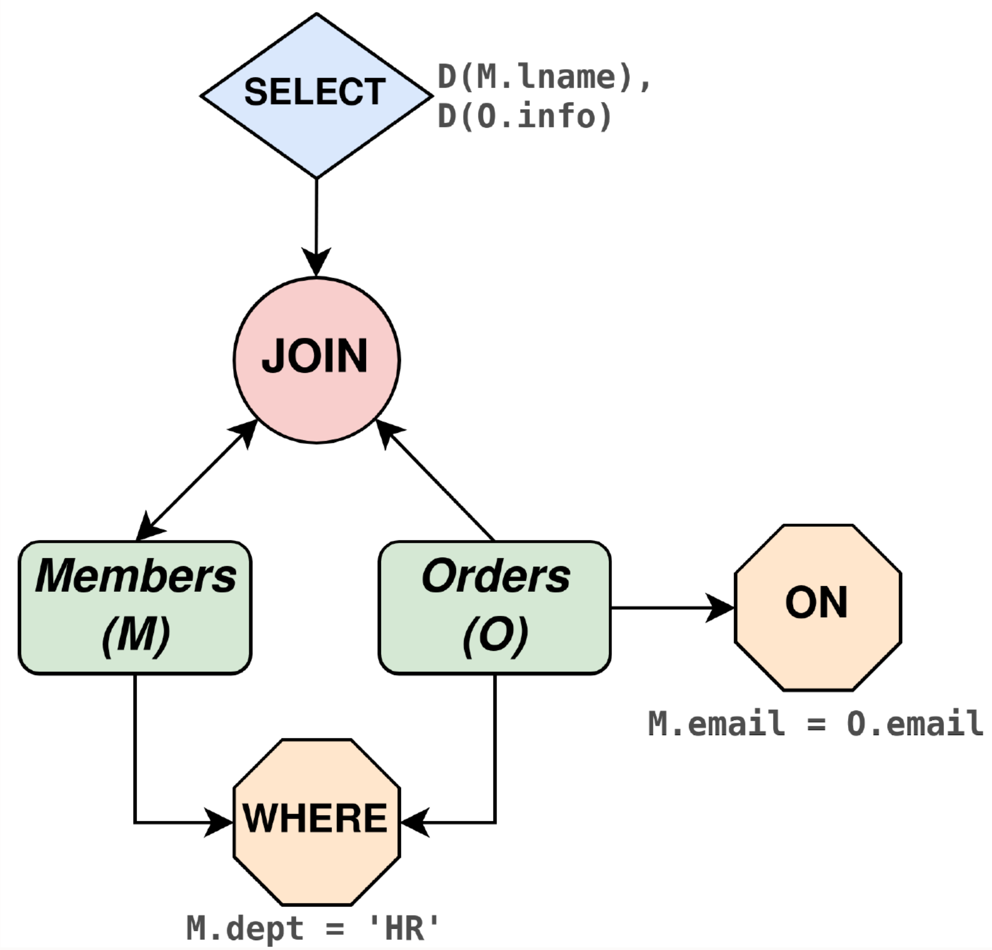
fname	lname	SSN	dept	email
John	Adams	XXXX	HR	j.adams@foo.com
Mary	Baker	XXXX	HR	m.baker@bar.com
Juan	Lopez	XXXX	Sales	j.lopez@foo.com
Amir	Zafar	XXXX	Sales	a.zafar@bar.com
Kira	Patel	XXXX	CEO	k.patel@baz.com

Computing Disclosure Using a Structural Graph

Reachability reveals accessed columns and predicates

Example query:

```
SELECT M.lname, O.info FROM members AS M
    LEFT JOIN orders AS O
    ON M.email = O.email
    WHERE M.dept = ‘HR’;
```



Computing Disclosure Using a Structural Graph

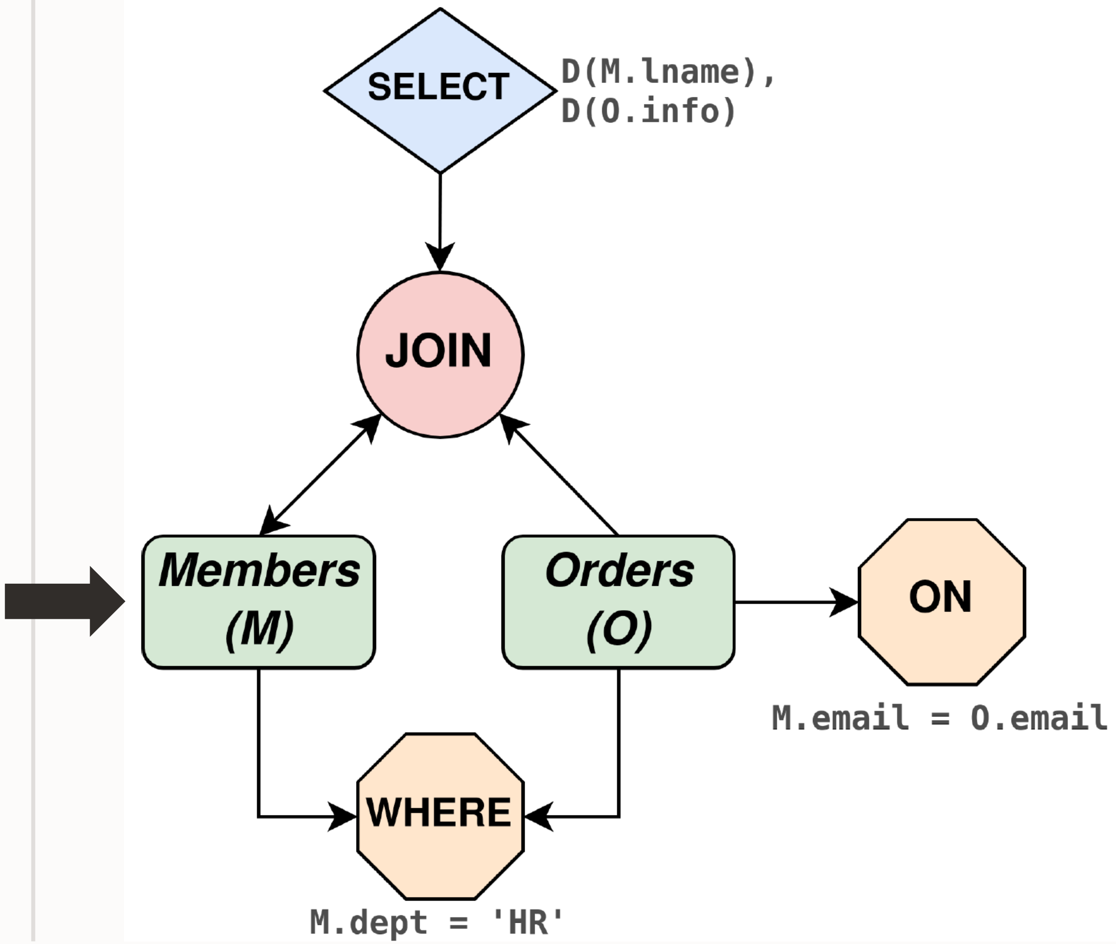
Reachability reveals accessed columns and predicates

Example query:

```
SELECT M.lname, O.info FROM members AS M
    LEFT JOIN orders AS O
    ON M.email = O.email
    WHERE M.dept = ‘HR’;
```

M.lname disclosure:

$$d_{M.lname} \rightarrow \left\langle \begin{matrix} \{M.dept\}, \\ M.dept = "HR" \end{matrix} \right\rangle$$



Computing Disclosure Using a Structural Graph

Reachability reveals accessed columns and predicates

Example query:

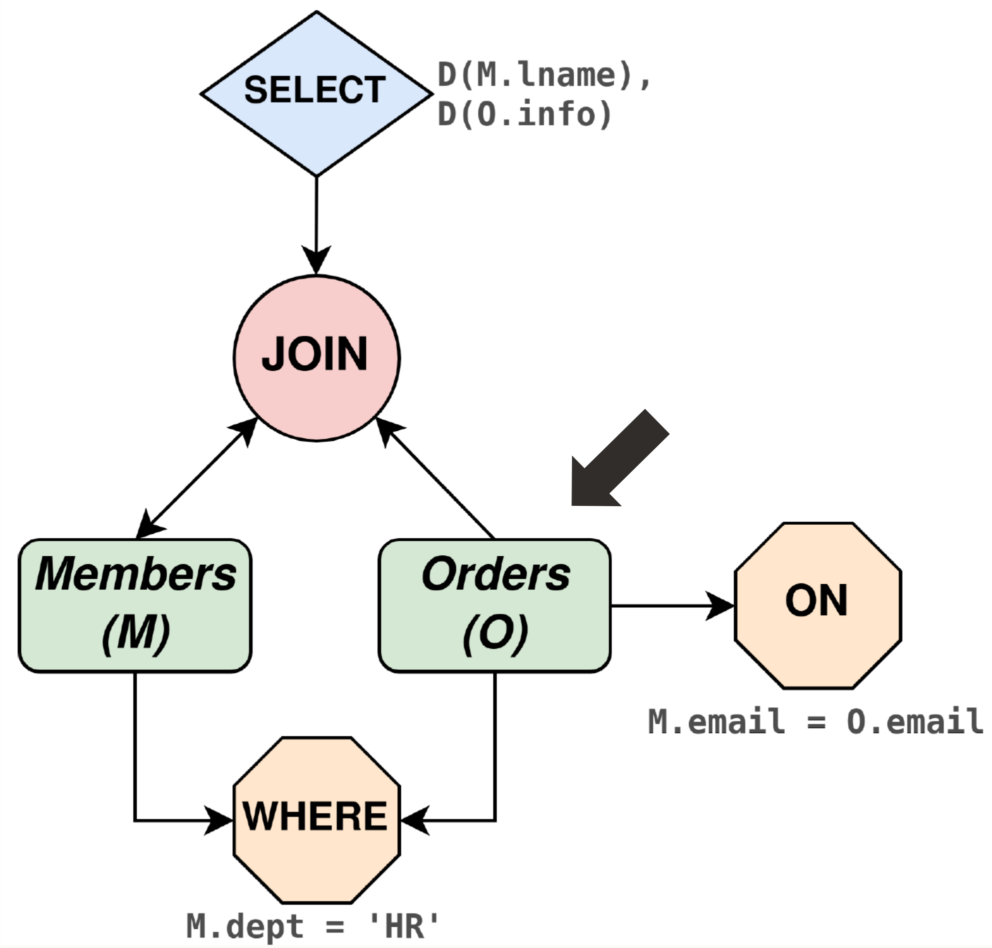
```
SELECT M.lname, O.info FROM members AS M
    LEFT JOIN orders AS O
    ON M.email = O.email
    WHERE M.dept = ‘HR’;
```

M.lname disclosure:

$$d_{M.lname} \rightarrow \left\langle \begin{matrix} \{M.dept\}, \\ M.dept = "HR" \end{matrix} \right\rangle$$

O.info disclosure:

$$d_{O.info} \rightarrow \left\langle \begin{matrix} \{O.email, M.email, M.dept\}, \\ M.email = O.email \wedge M.dept = "HR" \end{matrix} \right\rangle$$



Using graph reachability to derive information disclosure is appealing in its simplicity and works well with simple SQL queries, commonly observed in open-source codebases.

Since our submission, we have discovered that industrial codebases often use advanced SQL features for which information flow is inconvenient to model using a graph abstraction.

We are now exploring alternative strategies, inspired by abstract interpretation, that will provide us with the flexibility required to handle industrial usage of SQL.

francois.gauthier@oracle.com

Senior Principal Researcher

Oracle Labs Australia

References

1. https://github.com/SEC642/modsec/blob/master/rules/base_rules/modsecurity_crs_41_sql_injection_attacks.conf

2. Ceccato, Mariano, et al. "SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities." *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*. 2016.

3. Medeiros, Ibéria, et al. "SEPTIC: detecting injection attacks and vulnerabilities inside the DBMS." *IEEE Transactions on Reliability* 68.3 (2019): 1168-1188.

4. Jahanshahi, Rasoul, Adam Doupé, and Manuel Egele. "You shall not pass: Mitigating sql injection attacks on legacy web applications." *Proceedings of the 15th ACM Asia Conference on Computer and Communications security*. 2020.

In a Nutshell: Disclosure-Based SQLi Detection

A query is accepted if it discloses no more information than training queries

Training queries:

- `SELECT lname, fname, email FROM members WHERE dept='HR';`
- `SELECT dept, `**`email`**` FROM members WHERE lname='Baker' OR dept='Sales';`

Decomposition into $\langle \boldsymbol{Access}, \boldsymbol{Predicate} \rangle$ for email:

- $email \rightarrow \langle \{dept\},\ dept = \text{HR} \rangle$
- $email \rightarrow \langle \{lname, dept\},\ lname = \text{Baker} \vee dept = \text{Sales} \rangle$

Generalization into $\langle \boldsymbol{Access}^{+}, \boldsymbol{Predicate}^{+} \rangle$ for email:

- $email \rightarrow \langle \{lname, dept\},\ lname = \alpha \vee dept = \beta \rangle$
- α and β are abstractions of the concrete values.

Accept incoming $\boldsymbol{email} \rightarrow \langle \boldsymbol{A_{email}}, \boldsymbol{P_{email}} \rangle$ if:

- $A_{email} \subseteq \{lname, dept\}$
- $P_{email} \Rightarrow\ lname = \alpha \vee dept = \beta$

