MySQL Record Carving

Esan Wit¹  Leendert van Duijn¹
Supervisor: Kevin Jonkers²

¹SNE University of Amsterdam
²Fox-IT

February 5, 2014
Introduction

Why:

- No easy option available
- Recover deleted records
- Recover damaged or corrupted records
- Mailing lists said that it was very difficult to recover any sensible data
  - Challenge accepted
Existing Work

- Databases leave data behind on deletion/updates, Stahlberg et al.
- Recovery from SQLite, Pooters et al.
  - Template matching
- Recovery from alternative sources, Frühwirt et al.
  - Logs, replication files, temporary tables
- Percona LLC has a tool for data recovery from InnoDB tables
  - Template matching
  - Tricky setup
  - Finds many false positives
- Proving database integrity via index properties, Kieseberg et al.
To recover deleted records from MySQL.

- What data remains after deletion of a record?
- What methods exist for recovering, parts of, this data?
- Can this be extended to cover more general damage or different databases?
- How do the differences between database systems relate to record recovery?
### Basic layout

<table>
<thead>
<tr>
<th>Record 1</th>
<th>Record 2</th>
<th>Record 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Record 4</td>
<td>Record 5</td>
</tr>
<tr>
<td>Deleted record</td>
<td>Record 7</td>
<td>...</td>
</tr>
</tbody>
</table>

**Table:** Generic layout of records

- MyISAM whole file
- InnoDB leaf pages only
Deleting a record

<table>
<thead>
<tr>
<th>Original</th>
<th>After deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00 00 00 03 00 63 01 08 fe 02 00 00 00 12 4c</td>
<td>........c........L</td>
</tr>
<tr>
<td>65 65 6e 64 65 72 74 20 76 61 6e 20 44 75 69 6a</td>
<td>eendert van Duij</td>
</tr>
<tr>
<td>6e 18 4c 65 65 6e 64 65 72 74 2e 76 61 6e 44 75</td>
<td>n.Leendert.vanDui</td>
</tr>
<tr>
<td>69 6a 6e 40 6f 73 33 2e 6c 61 6e 64 65 72 74 2e 6c 36</td>
<td><a href="mailto:ijn@os3.nl">ijn@os3.nl</a>(5baa6)</td>
</tr>
</tbody>
</table>

Fragment of MyISAM data file
Deleting a record cont.

MySQL Record Carving

Esan Wit, Leendert van Duijn
All records have a similar structure:

```
header | field1 field2 ... fieldN
```

Header content and length depend on various properties:

- Record state
- Engine, row format
- Null fields
- Variable length fields
Template matching

- Attempt to parse data
- Be careful and strict
- Any misalignment will ‘corrupt’ output
- Some types lack visible boundaries
Template matching

- Sliding window template matching
- Data type validation
- Extensible data validation
for location in candidates
   for template in templates
      record = parse(location, template)
      if(success(record))
         score = validate(record)
         if(score > threshold)
            results.add(record)
            candidates.add(suggestion(record.length))
Early validators

An early validator can do elimination, when no full record is available yet e.g.

- Enum is invalid (i.e. it’s value is out of bounds)
- A string contains invalid characters according to its character set
- A record(header) is (not) marked as DELETED
- A record(header) is marked as RESERVED

Correct template design and strict early validators can limit false positives, while some match any position found.
A validator scores a row, based on field content e.g.

- String field `Name` contains common English letters
- String field `Email` looks like a valid email address
- Timestamp field `Last login` is within a realistic range
- Field `Last login` > `Registration date`
Results and observations

**Simple test on InnoDB (98306 matches attempted, 12 matched)**

<table>
<thead>
<tr>
<th></th>
<th>Present</th>
<th>Deleted</th>
<th>False pos.</th>
<th>False neg.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validated</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Expected</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

**InnoDB world (507922 matches attempted, 9193 matched)**

<table>
<thead>
<tr>
<th></th>
<th>Present</th>
<th>Deleted</th>
<th>False pos.</th>
<th>False neg.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validated</td>
<td>4042</td>
<td>100</td>
<td>88</td>
<td>25</td>
<td>4142</td>
</tr>
<tr>
<td>Expected</td>
<td>3979</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>4079</td>
</tr>
</tbody>
</table>

- Using validators we were able to reduce false positives.
- Template and validator quality determines overall effectiveness
Features of PoC

- Currently supported engines
  - InnoDB Antelope file format, Compact row format.
  - InnoDB Antelope file format, Redundant row format.
  - MyISAM Fixed row format
  - MyISAM Dynamic row format
- Early field level validation
- Row level validation
- Many different field types
  - datetime, decimal, varchar, set, etc.
Conclusion

- After executing a DELETE data remains partially recoverable
- Template matching can be used to recover this data
- Validation can be a powerful tool in eliminating false positives
Future work

- Other database systems
- Generic templates
- Split records
- Validators
- Machine learning
Questions?
Sources


Ivo Pooters and Pascal Arends and Steffen Moorrees, *Extracting SQLite records - Carving, parsing and matching*, 2011,

