Vulnerablities in the fee-payment gateway

Bharadwaj

October 28, 2018

Contents

1 Session cookie exploit

On the fee-payment portal, when cookie sent by the website, if deleted manually by the user and a hard-refresh is done the **debug info/ backtrace** is made public pointing to the following:

1. Raw sql query.
2. Location on disk where the file is stored.

This makes it easier for the attacker to identify the location of the files stored.

2 SQL injection attack

The image above is what I could reproduce by deleting the session cookie.

The reason this could spring up a possible SQL injection attack is the following query:
sql = "Select * from lfdcpay where admno=" + Session["amdn"].ToString()

Figure 1: Debug mode sql error

for a naive eye this might look like a normal query concatenating information colleted in the form.

But concatenating in this way can prove to be dangerous.

I will now provide a context for why this is dangerous:

2.1 Example

cmd.CommandText = "Insert INTO workers Values (" + User.Identity.Name + "," + WorkerName.Text + "," + GetUniqueWorkerKey() + ");";

consider the above query and the reasons to not follow this type of queries:

Short answer: building queries by concatenating strings usually allows SQL injection.

Imagine that someone tries to create a user with the name "Bob, Joe, 12345); DROP TABLE workers; –". You end up building the query "Insert INTO workers Values(Bob, Joe, 12345); DROP TABLE workers; –name, 34345236);" Bye-bye database. SQL injection can also lead to queries returning data that they shouldn't, such as password tables. Any time that you have SQL injection, just assume that you're allowing arbitrary third parties to issue arbitrary commands to your database.

3 HTTPS not available

Though the payment process is routed thorugh a HTTPS protol; the form where we enter our admission number and phone number is not HTTPS routed, instead it follows classic HTTP protocol.

Information entered this way can be stolen easily as the communicaton between the server and the client is not encrypted. A Man-in-the-middle attack

is easy to perform if we manipulate the HTTP requests made to the server.

4 Immediate actions to be taken

4.1 Turn off debug mode

The figure 1 shows **debug mode** for any errors generated during any request made to the server. However, though this might seem to be an easy way to deal with exceptions that occur when any request acts as not intended, it is to be used while in **development mode only**; *when in production mode, one should never leave such debug information available to the public.* The reason is that anybody with sufficient bruteforcing skills can enter into the server and can cause irreparable damages.

4.2 Alternatives to concatenating queries

An alternate method has to be implemented to protect from possible SQL injection attack.

4.3 HTTPS implementation

HTTPS protocol should be be made defacto protocol for the entire website.

Let's Encrypt is one such CA(Certification Authority) that issues *free* SSL/TLS certificates.