CAS CS 357

In-Class Note 3

D

1. SQL Injection Attack
2. D
3. Vulnerability 🡪 the attacker can interact with the database using SQL code
4. Exploit 🡪> attacker causes damage (e.g. steal a password from the user)
5. SQL injection happens when input is processed as SQL code

Ex) java code

String query = “SELECT account\_balance FROM user\_data WHERE user\_name = ’ + request.getParameter(“customerName”);

1. The “request.getParameter(“customerName”);” section is vulnerable and would allow attacker to inject code into the query that would be executed by database
2. Preventing SQL Injection
3. Never build SQL command by yourself
4. Rather, use parameterized/prepared SQL or ORM framework
5. In other words, don’t write your own parser, since you will mess up
6. Prepared statements
7. Defense option 1: prepared statements: the use of prepared statements with variable binding is how all developers should first be taught how to write database queries since they are simple to write and easier to understand
8. It forces the developer to first define all SQL code and pass each parameter to query later
9. This method treats code as a text
10. This style allows database to distinguish between code and data, regardless of the input
11. Example in java

String query = “SELECT account\_balance FROM user\_data WHERE user\_name = ?”; -🡪 leaving the question mark is te significant part

try {

OleDbCommand command = new OleDbCommand (query, connection);

command.Parameters.Add(new OleDbParameter(“customerName”, CustomerName Name.Text)); 🡪 the variable Name.Text is converted into parameter and therefore, SQL attacks through quotation (‘’) are not possible since the value entirely is viewed as a string variable

…

1. Another example in java

String custname = request.getParameter(“customerName”); 🡪 get connection from a form

String query = “SELECT account\_balance FROM user\_data WHERE user\_name = ?”;

PreparedStatement pstmt = connection.prepareStatement(query);

pstmt.setString(1, custname);

ResultSet results = pstmt.executeQuery();

1. Through using this method, we can view vulnerable codes such as “Robert’;Drop table; --” as a simple text and not code, that can possibly attack the SQL through the usage of single quotation and therefore making ourselves vulnerable to a possible SQL injection
2. Input list validation (allow lists) 🡪 not recommended
3. Allow lists are lists that the programmer creates (possibly using the switch case method in java) where the programmer only accepts certain types of results in a query such as only accepting “Bob”, “Chris”, and “Jason” for the field “name”
4. Example in java:

String tableName;

switch(PARAM):

case “Value1”: tableName = “fooTable”;

break;

case “Value2”: tableName = “barTable”;

break;

default : throw new InputValidationException(“unexpected value provided” + “ for table name”);

1. Here, we restrict the table name to only fooTable and barTable to protect from possible SQL injection attack to the table. If other value other than the two table names are given (or an unexpected name not listed in the switch-case code is written, an error will appear, protecting the code from a SQL attack using quotation
2. Other example:

String query = “SELECT users FROM ” + tableName + “WHERE name = ??”

1. The ?? that we later provide should match with one of the options in the switch-case code in order for the query to run
2. However, this is a symptom of poor design and full rewrite should be considered
3. Not recommended because depending on which case it undergoes in the list, things can react differently (things can go wrong and not intended)
4. In the example above, there are only two possible table names but in reality, there can be many more options and simply writing code for each case can lead to errors or things can react differently the way not intended (not the same process goes through bootable and footable, causing errors or leading to different solutions that are not intended, eventually)
5. In addition, if you allow allowlist, emails that look like they come from you or someone else in your company, but are really sent by spammers, get delivered to your Inbox read to do harm
6. URLS
7. Given a url, <http://standford.edu:81/class?name=cs155#homework>, we can identify different parts of the url
8. http: 🡪 protocol (prescribed order and syntax for presenting information or an agreed set of rules devices follow in order to communicate with each other or standardized set of rules for formatting and processing data)
9. standford.edu 🡪 hostname (name of the server)
10. 81🡪Port (folder for each class)
11. class🡪path (complete location or name of where computer, file, device, or web page is located)
12. name=cs155 🡪query (question or request for information expressed in formal manner – information comes from database, if not correctly parsed, adversaries can enter and do all sorts of stuffs)
13. #homework 🡪 fragment (condition in which the contents of single file are stored in different locations on the disk rather in contiguous space)
14. HTTP Request (what you request to the website)

Ex)

GET /index.html HTTP/1.1 🡪 Get is method, index.html is file, and 1.1 is HTTP version

Accept: image/gif, image/x-bitmap, image/jpeg, \*/\* 🡪 Headers

Accept-Language: en

Connection: Keep-Alive

User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)

Host: [www.example.com](http://www.example.com)

Referer: <http://www.google.com?q=dingbats> (shows where you are coming from (privacy issue) – telling the server what the user was looking at before coming to the website – revealing to the server you came from)

1. In GET, there are no side effects
2. In POST, possible side effect exists
3. HTTP Response

Ex)

HTTP/1.0 200 OK 🡪 1.0 is HTTP version, 200 is status code (like 404), OK is reason phrase

Date: Sun, 21 Apr 1996 02:20:42 GMT

Server: Microsoft-Internet-Information-Server/5.0 🡪 Microsoft-Internet-Information-Server/5.0 is header

Connection: keep-alive

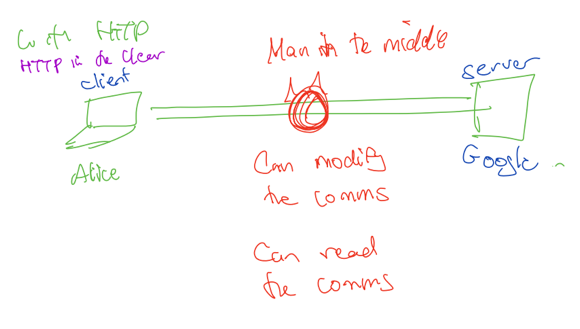
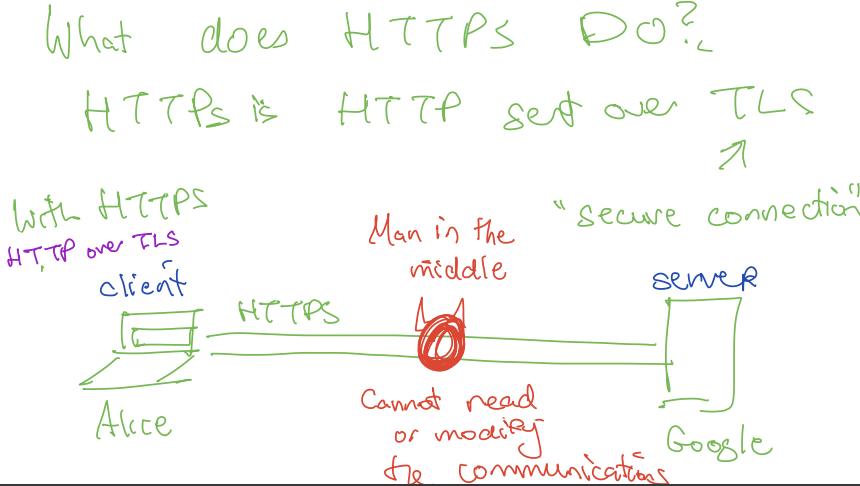
Content-Type: text/html

Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT

Set-Cookie: … 🡪 When you visit website, web server transfers small packet of data to your device’s browser, known as cookie. They are small files that include unique identifier that web servers send to browsers. It’s a way for a website to remember you, preferences, and habits online. When get response back from server, the server might return a cookie to the user (cookie is what makes browser maintains state with the website)

Content-Length: 2543

<HTML> Some data … blah </HTML> 🡪 Data

1. Http:// versus https://
2. http:// is more dangerous and more vulnerable. It sends HTTP in the clear (adversary can see them or can alter communications
3. 
4. https:// means we are sending https:// over TLS – protect against network attacks
5. 
6. Ex) Http, you can steal card number
7. Https is an encrypted connection between the user and the browser
8. In https, HTTP requests and responses are encrypted with the session keys (private and public keys) so that anyone who intercepts communications can only see random string of characters, making it difficult for the man in the middle to read, modify, and communicate the data