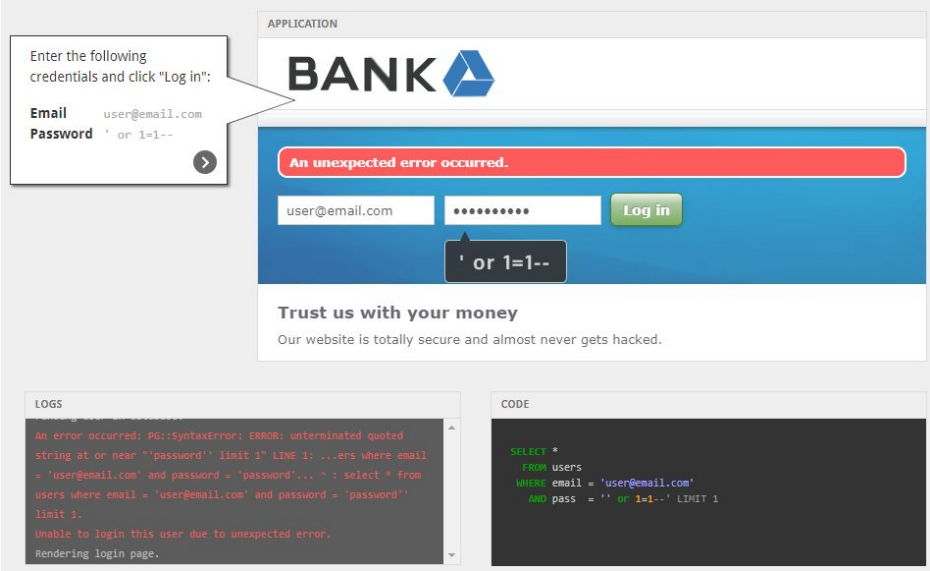
Common
Web Vulnerabilities

By: Mihailo Kalinic

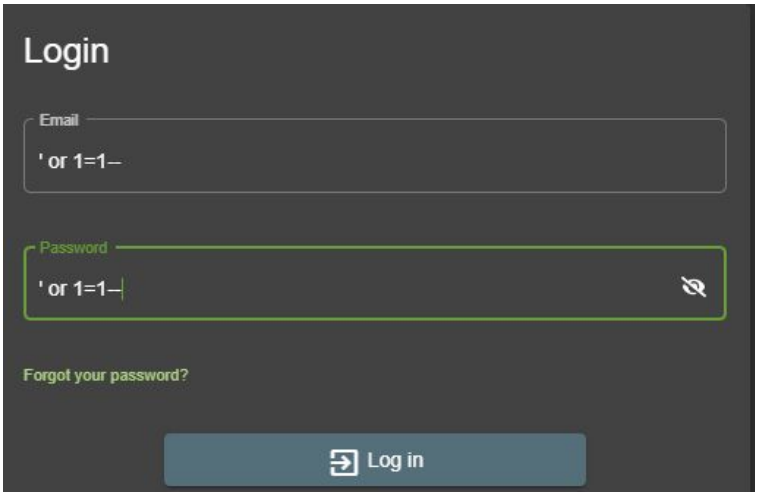
Notes are sourced from hacksplaining and portswigger.net

SQL Injection

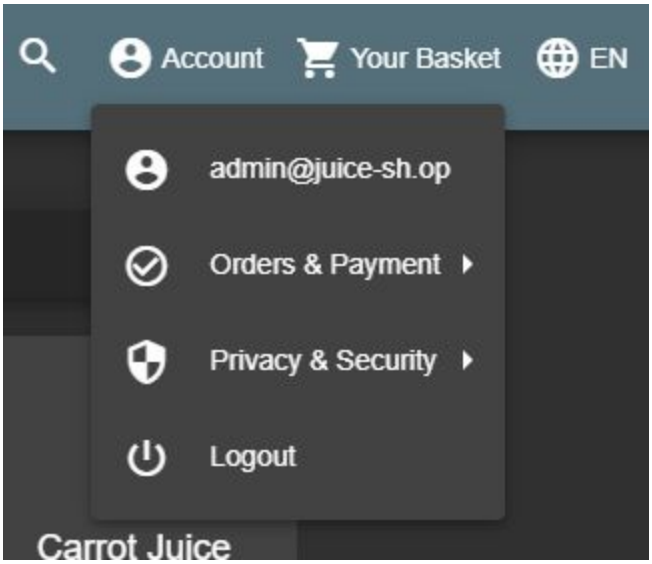
- Injection attacks occur when maliciously crafted inputs are submitted by an attacker, causing an application to perform an unintended action.
- One of the most common attacks on the internet.



The -- characters allowed the database to ignore the rest of the SQL statement and allowed us to be authenticated through boolean logic (OR) as 1=1 has tricked the database and allowed us to be authenticated into the account.



This can also be used in the user prompt, and in OWASP Juice shop it logs us into the admin account of the webpage, this is because the first account on the SQL table is generally the admin account.



Prevention Measures

1) **Parameterised Statements**

Parameterised statements make sure that the parameters (i.e. inputs) passed into SQL statements are treated in a safe manner.

Make sure that statements are passed to the database separately, which allows the driver to correctly interpret them.

2) **Object Relational Mapping (ORM)**

Use Object Relational Mapping (ORM) frameworks to make the translation of SQL result sets into code objects more seamless. ORM tools often mean developers will rarely have to write SQL statements in their code – and these tools thankfully use parameterized statements under the hood.

ORM frameworks allow you to construct SQL statements, or fragments of SQL statements, when more complex operations need to be performed on the database.

3) **Sanitizing Inputs**

Sanitizing inputs is a good practice for all applications. In our example the user supplied a password as `' or 1=1--`, which looks pretty suspicious as a password choice.

Make an effort to reject inputs that look suspicious out of hand, while taking care not to accidentally punish legitimate users.

Cross Site Scripting (XSS)

- Sites which allow users to add content can be susceptible to XSS, as attackers can inject malicious javascript. These attacks commonly use XSS to steal another user's cookie and permit session hijacking.
- XSS vulnerabilities permit a malicious user to execute arbitrary chunks of JavaScript when other users visit your site.
- XSS is the most common vulnerability.

What could a determined hacker do when exploiting a XSS vulnerability?

XSS allows arbitrary execution of JavaScript code, so the damage that can be done by an attacker depends on the sensitivity of the data being handled by your site. Some of the things hackers have done by exploiting XSS:

- **Spreading worms on social media sites.** Facebook, Twitter and YouTube have all been successfully attacked in this way.
- **Session hijacking.** Malicious JavaScript may be able to send the session ID to a remote site under the hacker's control, allowing the hacker to impersonate that user by hijacking a session in progress.
- **Identity theft.** If the user enters confidential information such as credit card numbers into a compromised website, these details can be stolen using malicious JavaScript.
- **Denial of service attacks and website vandalism.**
- **Theft of sensitive data**, like passwords.
- **Financial fraud** on banking sites.

Prevention

1) Escaping Dynamic Content
2) Whitelisting Values
3) Sanitizing HTML

Directory Traversal

Directory Traversal attacks, exploit the URI specifically the URL of websites whose file paths are allowed to be seen with unsafe interpretation. This mainly works as servers do not validate file paths and hence this will happen (if web developer has not sanitized web files to display URL properly).

These attacks tend to occur in older technology stacks, which map URLs too literally to directories on disk, this can be seen above with the apache server which still implements a LAMP stack for its website.

Attackers find this vulnerability through using google dorks as the following;

```
site:<yourdomain.com> inurl:file=
```

The results which are returned from the index of search an attacker can weaponize.

Prevention

1) Use a Content Management System
2) Segregate Documents
3) Sanitize website contents properly
4) Use restricted permissions

File Upload Vulnerabilities

FUV is an easy way attackers can inject malicious code into applications. This attack is favoured by hackers as it requires sites to take a large chunk of data and write it to disk space. When writing it allows attackers to put malicious scripts on the servers which can then comprise the web server.

You need to ensure uploaded files are kept at arm's length until they are fully secured, or else you risk creating an easy route to having your systems compromised.

Protection

1) Segregate your Upload
2) Upload files should not have an executable permission
3) Rename files
4) Virus scanner