Networking OWASP 3 Topics

1. Injection
2. Cross site scripting
3. Insufficient logging and monitoring

Injections example:

SQL Injections are used to inject code into a database which allows for it to preform many action these being: Retrieving hidden data, UNION attacks, examining the database.

Hidden Data:

Consider a shopping application that displays products in different categories. When the user clicks on the Gifts category, their browser requests the URL:

https://insecure-website.com/products?category=Gifts

This causes the application to make an SQL query to retrieve details of the relevant products from the database:

SELECT \* FROM products WHERE category = 'Gifts' AND released = 1

This SQL query asks the database to return:

all details (\*)

from the products table

where the category is Gifts

and released is 1.

The restriction released = 1 is being used to hide products that are not released. For unreleased products, presumably released = 0.

The application doesn't implement any defenses against SQL injection attacks, so an attacker can construct an attack like:

https://insecure-website.com/products?category=Gifts'--

This results in the SQL query:

SELECT \* FROM products WHERE category = 'Gifts'--' AND released = 1

The key thing here is that the double-dash sequence -- is a comment indicator in SQL, and means that the rest of the query is interpreted as a comment. This effectively removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed, including unreleased products.

Going further, an attacker can cause the application to display all the products in any category, including categories that they don't know about:

https://insecure-website.com/products?category=Gifts'+OR+1=1--

This results in the SQL query:

SELECT \* FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1

The modified query will return all items where either the category is Gifts, or 1 is equal to 1. Since 1=1 is always true, the query will return all items.

From: <https://portswigger.net/web-security/sql-injection>

\*Very cool and interesting method, but how prevalent is this vulnerability?

\*don’t understand UNION attacks that well

Cross site scripting:

Client side script execution which is a common vulnerability however it is easy to find and fix.

Suppose there's a URL on Google's site, http://www.google.com/search?q=flowers, which returns HTML documents containing the fragment

<p>Your search for 'flowers' returned the following results:</p>

i.e., the value of the query parameter q is inserted into the page returned by Google. Suppose further that the data is not validated, filtered or escaped.

Evil.org could put up a page that causes the following URL to be loaded in the browser (e.g., in an invisible<iframe>):

http://www.google.com/search?q=flowers+%3Cscript%3Eevil\_script()%3C/script%3E

When a victim loads this page from www.evil.org, the browser will load the iframe from the URL above. The document loaded into the iframe will now contain the fragment

<p>Your search for 'flowers <script>evil\_script()</script>'

returned the following results:</p>

Loading this page will cause the browser to execute evil\_script(). Furthermore, this script will execute in the context of a page loaded from [www.google.com](http://www.google.com).

Insufficient logging and monitoring

An open source project forum software run by a small team was hacked using a flaw in its software. The attackers managed to wipe out the internal source code repository containing the next version, and all of the forum contents. Although source could be recovered, the lack of monitoring, logging or alerting led to a far worse breach. The forum software project is no longer active as a result of this issue.