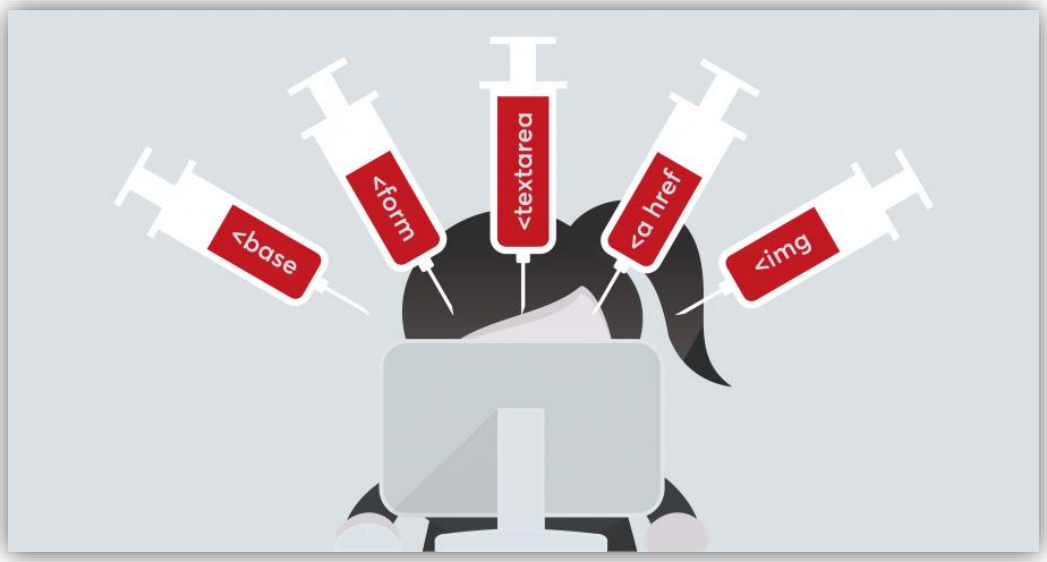
HTML Injection



What is HTML?

HTML is the language that determines how application data (like a products catalog) gets presented to users in their web browser.

This language contains visualization commands, like the colour of the page's background and the size of embedded pictures. It also contains links to other web pages, and additional commands intended for the user's browser.

On modern interactive websites, the content of a web page often reflects the processing results of previous user actions.

If user input is not validated and the application is vulnerable, an attacker could create and send input to the application that allows a piece of HTML code to be injected into the HTML content in the application's response.

HTML Injection:

HTML injection is a web vulnerability that lets an attacker inject malicious HTML content into legitimate HTML code of a web application.

HTML Injection is an attack that is similar to Cross-site Scripting (XSS). While in the XSS vulnerability the attacker can inject and execute Javascript code, the HTML injection attack only allows the injection of certain HTML tags.

When an application does not properly handle user supplied data, an attacker can supply valid HTML code, typically via a parameter value, and inject their own content into the page.

The Malicious user(attacker) sends HTML code through any vulnerable field with the purpose to change the website's design or any information that is displayed to the user.

As a result, the user may see the data that was sent by the malicious user. Therefore, in general, HTML Injection is just the injection of markup language code to the document of the page.

Because it takes advantage of a trust issue and a code-based vulnerability, this attack is usually employed in tandem with social engineering.

HTML injection is planned mainly because of two goals:

- To modify the website appearance so that the website's reputation is tarnished.
- To snatch the identity of someone authorized.

How Does HTML Injection Work?

The attack is executed using the links and data input fields of the targeted website.

To bring this attack into action, steps as follows:

1. Hackers begin by locating websites with weak HTML coding and injecting HTML. (HTML injection is commonly utilised in website elements like search bars, comments, and contact forms.)

2. Functionality of this attack can be done by with the help of questionnaire form (which is allows web designer to understand or gain insight into intended user's needs and wants to ensure that the final product delivered by web designers is an exact match.

   Then hacker compromises the website and then add HTML code snippets (a small portion of source code in HTML) that lead the intended user(victim) to believe website intended and force them to download malicious software.

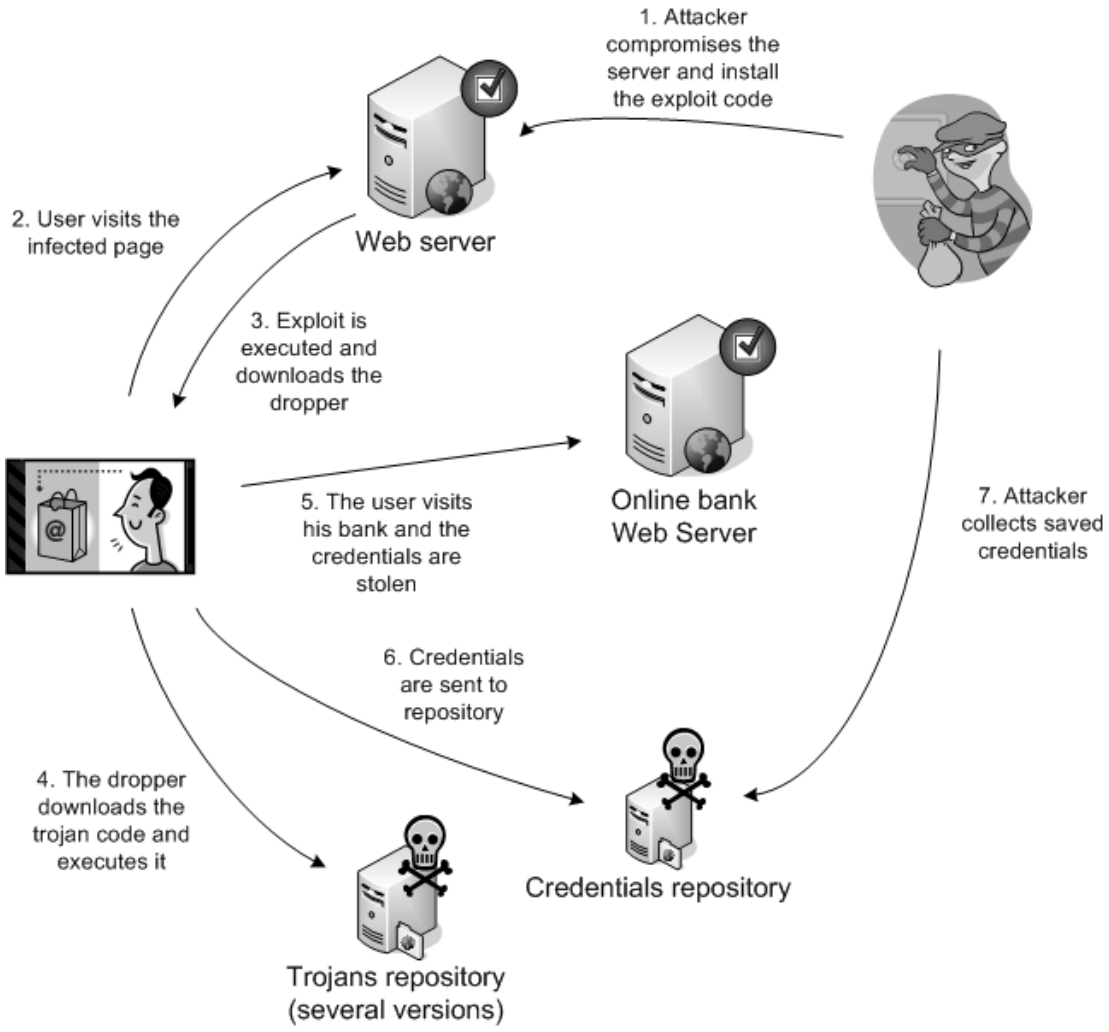
3. a user answers these questionnaires on any of these websites, they share information such as concerns, names, email addresses, and phone numbers.

4. Upon submission of questionnaire form, an acknowledgment message is shared instantly to users, The corresponding code for this message will be:

   ```
   var user_name=location.href.indexOf("user=");
   document.getElementById("Thank you for filling our questionnaire").innerHTML=" Thank you for filling our questionnaire, "+user;
   ```

5. Since the code is simple to modify and highly susceptible, hackers will insert an HTML injection into the code.

Showing another simple way of how HTML Injection performed through image



Another possible attack scenario is demonstrated below:

- Attacker discovers injection vulnerability and decides to use an HTML injection attack
- Attacker crafts malicious link, including his injected HTML content, and sends it to a user via email
- The user visits the page due to the page being located within a trusted domain
- The attacker's injected HTML is rendered and presented to the user asking for a username and password
- The user enters a username and password, which are both sent to the attacker’s server

Types of HTML Injection:

1. ## Stored HTML Injection:

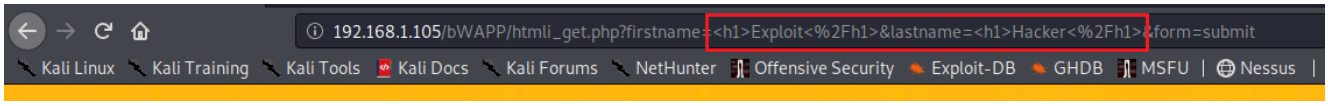
   A stored HTML attack also known as "Persistence" occurs when a malicious script is injected into a web application and then permanently stored inside the application/Web server. The web server then dumps the malicious script back out to the user when the user accesses the injected webpage.

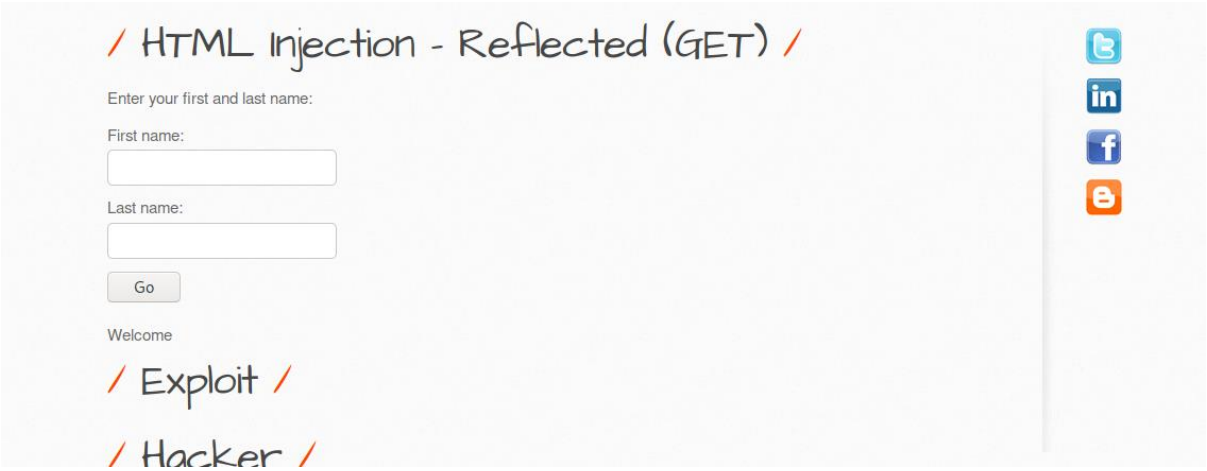
2. ## Reflected HTML Injection:

   It occurs when an application receives data in an HTTP request and unsafely incorporates that data into the immediate response. Reflective HTML injection vulnerability allows an attacker to inject malicious code into a website, but they will only be visible to the current user that is neither stored in the database nor visible to anyone.

   This can again be divided into more types:

   I. Reflected GET HTML Injection:
   this Injection occurs when our input is being displayed (reflected) on the website.





GET method is used to retrieve information from the given server using a given URI. If the attacker noticed the vulnerability, they could lead the phishing attacker as they could create a phishing page through linking elements to the URL and gain vicitm credentials.

II. Reflected POST HTML Injection:

This occurs when a malicious HTML code is being sent instead of the correct POST method parameters.

For example, we have a login form that is vulnerable to HTML attack. Data typed in the login form is being sent using the POST method. Then, if we type any HTML code instead of the correct parameters, then it will be sent with POST method and displayed on the website.

To perform the Reflected POST HTML attack, it is recommended to use a special browser plugin that will fake the sent data. One of them is the Mozilla Firefox plugin “Tamper Data”. The plugin takes over the sent data and allows the user to change it. Then changed data is being sent and displayed on the website. Attackers may not lead a damaging attack if the vulnerability is in same place, but the data is traveling through a post request. This vulnerability can be effect only the current user.

III. Reflected URL HTML Injection:

happens when HTML code is being sent through the website URL, displayed in the website and at the same time injected into the website's HTML document.
It is not necessary that you always find the HTML injection vulnerability in the input field, even some times the URL that is shown on the website may be unsafe or vulnerable.

How to Test Against HTML Injections?

When starting to test against a possible injection attack, a tester should first list out all the potentially vulnerable parts of the website that is All data input fields and Website link.

The manual tests could be performed as follows

When testing manually, a basic HTML code could be entered to see if the text would display, for example, if HTML Injection is possible. (It makes no sense to test with extremely complex HTML code; straightforward code might be sufficient to determine whether it is displayed).

For example, there may be simple tags with text:

```
<h1>HTML Injection testing</h1>
```

If an HTML code being saved somewhere is displayed, then the tester can be sure that this injection attack is possible. Then a more complicated code may be tried, to display the fake login form.

The HTML Injection Scanner is an additional solution. You could save a ton of time by having your system automatically scan for this attack. In contrast to other attacks, I would like to let you know that there aren't as many tools available for testing HTML Injection.

WAS Application (WebSphere Application Server):

WAS can be named quite a strong vulnerabilities scanner, as it tests with different inputs and does not just stop with the first fail.

It is helpful for testing, browser plugin “Tamper Data”, it gets sent data, allows the tester to change it and sends it to the browser.

Mitigation and Prevention HTML Injection:

Below mentioned are some viable ways to control the spread and risks of HTML injections.

1) Since the attack only targets unverified inputs and outputs, validating inputs and outputs is the first and most important step in preventing HTML injection.
2) Inspecting each input to see whether any reference of HTML or script code is present. To aid you in the process, there are numerous tools available.
3) Implementing good security testing procedures is another effective way to stop these attacks from spreading. To ensure that no website component is overlooked during testing, consider utilizing automated testing solutions.
4) Using a Web Application Firewall, or WAF, is an excellent mitigation strategy to use. Website owners can prevent users or hackers from changing the input codes by using WAF. In this manner, HTML codes are excluded from website inputs. To lessen this kind of attack, a content security policy, or CSP, can be used. But keep in mind that not every use case will be covered by this policy. As a result, you must also embrace additional strategies.

Comparison HTML Injection with other Attacks:

This attack will undoubtedly not be seen as dangerous as an XSS, JavaScript Injection, or SQL Injection attack, in contrast to the other potential attacks. It won't erase the entire database or take every piece of information out of it. It shouldn't be written off as unimportant, though.

the main purpose of this type of injection is changing the displayed website's appearance with malicious purpose, displaying your sent information or data to the final user. Those risks may be considered less important.

It should be mentioned, that for stealing other user's data, this type of attack is less frequently selected, as there are a lot of other possible attacks.

However, this is very similar to an XSS attack that steals a user's cookies or other users' identities. There is also his XSS attack based on HTML. Therefore, testing for XSS and HTML attacks are very similar and can be run together.

References

i. https://www.researchgate.net/figure/Real-HTML-injection-example_fig1_39556078
ii. https://www.imperva.com/learn/application-security/html-injection/
iii. https://www.acunetix.com/vulnerabilities/web/html-injection/
iv. https://www.wallarm.com/what/html-injection