Application of Unsupervised Learning for Detection Cross-site Scripting (XSS) Security Breaches

Paloma Symmonds

College of Engineering,
Embry-Riddle Aeronautical University
Daytona Beach, FL, United States

Ephraim Nielson

College of Technology and Computing
Utah Valley University
Orem, UT, United States

Ali Alharbi

School of Engineering and Computer Science
Oakland University
Rochester, MI, United States
Email: aoalharbi [AT] aokland.edu

Abdulaziz Alshammari

School of Engineering and Computer Science
Oakland University
Rochester, MI, United States

Mohamed A. Zohdy

School of Engineering and Computer Science
Oakland University
Rochester, MI, United States

Abstract—**Advancements of technology in the field of networking and computing bring with it a heightened importance for cyber security. Currently, legal forms of identification, financial information, and scientific data rely on this technology. Moreover, as more dependence on cloud computing, data-base storage, and online banking is applied keeping sensitive information secure is paramount. JavaScript Cross-site based attacks continue to be the most prevalent cause of compromised information. Here, we demonstrate the feasibility of using unsupervised learning algorithms to detect attacks as part of the Intrusion Detection System for malicious cross scripts with attacked web sites. Our contribution is domain-based, in order to track changes of the interaction. Profiles are made from web-crawled pages and parsed according to key scripting features. The detection is done when scripting deviates from the main clustering of those clean profiles data gathered**

Keywords-component; Cybersecurity, unsupervised neural networks, cyber-attacks, cross-site scripting, XSS, anomaly detection

I. INTRODUCTION

Web attacks have become one of the greatest causes of identity theft, stolen assets, and compromised information security. In 2015 alone, 9 large-scale breaches and close to 429 million exposed identities were reported [1]. Despite the severity of reported attacks, it is only a percentage of actual cases of compromised information.

Of all cyber-attacks, JavaScript based attacks have been shown to be one of the most prevalent [1, 2, 3]. JavaScript attacks include injection type of attacks such as cross-site scripting (XSS), drive-by downloads, and cross-site forgery requests (CSFR). Despite the severity of these attacks, most web programming languages do not provide a guarantee of safe data transfer to the client by default [4].

While several methods of scripting attack prevention systems continue to be developed, hackers continue to find new methods to bypass current systems. Even large, established companies with a high focus on security are shown to be susceptible to such attacks [1, 6, 7, 23, 24].

In the case of XSS, since no pre-defined characteristics exist, the method of detection is called "anomaly detection". Anomaly detection first examines what is deemed normal and then detects deviations from the normal behavior [8,9,22].

In our work, anomaly detection is sufficient for classifying XSS attacks under analysis techniques. Using this method, a database of web application profiles will be created from historical audited data for each domain tested. Subsequent profiles will be created over a period of time as interaction with the web application occurs. The generated new profiles that were marked as normal profiles will be compared with the set of existing profiles by using a machine learning technique (anomaly detection). If the profile deviates significantly from the profiles in the database, it is marked as an intrusive state. The webpage might be infected by XSS, data invasion, or data manipulation and need further analysis (Fig. 1).

While many methods of XSS detection have already been introduced, our methodology differs in a few instances from other works. Some of the differing characteristics are:

- Detection is not limited to untrusted behaviors. In other words, untrusted user's behavior will be detected as

well other users who are merely injecting inputs into the field's area.

- It is not assumed that certain strings are malicious based off previous trained networks. These assumptions will not be made because scripting methods for attacks are constantly evolving and could change.
- Anomaly detection is host-based rather than network-based. Host-based anomaly detection is more focused on analyzing and extracting the data from web applications directly.
- The intrusion detection process is carried out in the server for static analysis and classification. This process provides a quick response to untrusted behavior to be detected and prevent a breach.

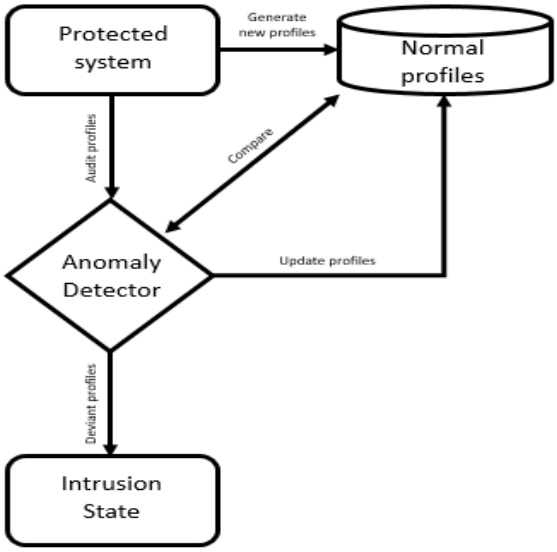

Figure 1. Anomaly detection flowchart

This paper is organized as follows. In section 2, we define the Cross-Site Scripting (XSS) and one of its categories. In other words, XSS has many types of attacks. Section 3 discussed how our methods is different among other techniques with examples. We will then introduce the details of our XSS detection scheme in Section 4. Section 5 describes the technique and evaluates performance of the mechanism, followed by discussing an example of Cross-Site Scripting and how the technique can be implemented in Section 6. The paper Conclusion and the acknowledgment in Section 7 and Section 8 respectively.

II. CROSS-SITE SCRIPTING

Cross-site scripting (XSS) can be defined as an attack vector caused by malicious code that is injected into trusted web applications using user input that is not properly validated or sanitized [10, 11, 12]. Sanitization is the removal of potentially harmful code from untrusted data [13]. XSS occurs when malicious scripts are sent by one user to another end user via a web application [12]. Other methods of implementing an injection attack vector include technologies that are supported by the browser being used (e.g. Flash, Active X, VBScript, XUL, QuickTime, etc…) [3, 14, 25].

There are three types of XSS: persistent or stored, non-persistent or reflective, and DOM-based. These different types of XSS fall into two categories of execution and storage: client side and server side. These definitions have been adopted by the research community since 2012 as seen in the Online Web Application Security Project (OWASP) [12].

A. *Persistent XSS*

The most damaging and costly XSS attack is the persistent, or stored attack, because the injected attack is permanently stored on the targeted server and is executed in the end user's browser any time that infected page is requested [12]. Because of this, the focus of our work will be to detect and defend against persistent or stored XSS.

To better understand persistent attacks, Fig. 2 is provided [15]. The attacker first finds a vulnerability in a web application where he/she can inject their code. These lines of code – that are normally written in JavaScipt – are then sent to the web application and directly embedded in the HTML structure of the website stored in its database [5, 10, 11, 12]. When the victim requests the page with the added injection, the content of that page is executed in the victim's web browser.

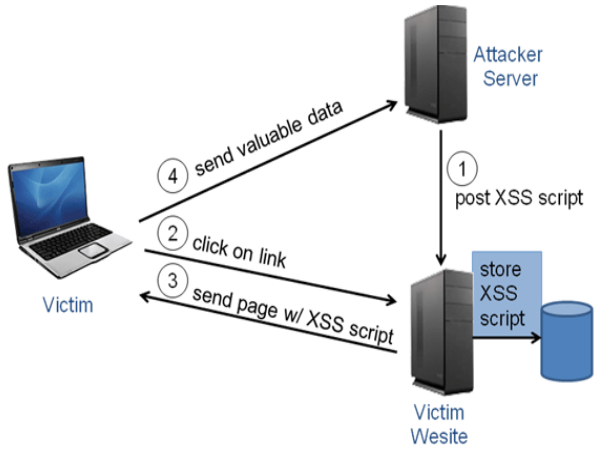

Figure 2. Persistent XSS example.

The purpose of the malicious code executed is not limited to a certain number of actions but is generally designed to compromise a victim's sensitive information or steal their assets. One example would be to steal a victim's session cookie and send it to the attacker. This would allow the attacker to hijack the victim's session and steal personal information as well as take over that account [12]. Other examples would be web redirects to a malicious web page, disclosing a victim's files, installations of Trojan horses, or modification of information [12]. In the case of social networking XSS, the code executed could also act as a worm and propagate throughout a victim's page and infect those who come in contact with them [15].

III. DISCUSSION

Other methods have been shown to be effective for classification in areas where the untrusted areas of code can be identified; however, using only these methods will eventually be insufficient to identify attacks because:

- Injections do not always occur in permissible areas and can be injected elsewhere in the HTML structure of the web application [3, 14].
- Training data for the supervised neural network needs to be up to date and routinely trained to keep up with performance [17].
- Supervised neural networks can classify malicious code as benign if intruders continually work below the tolerance threshold and train the network to misclassify increasingly abnormal code [18].

For example, while obfuscation is used in many malicious attacks, the same obfuscation occurs in legitimate benign code [16, 19]. A high number of eval() functions is also seen as a characteristic of XSS; however, benign code uses the eval() function for many legitimate purposes [14]. Another characteristic historically viewed as typical of malicious scripting would be a high keyword to word ratio and the number of functions [16]. These keywords can be found on the w3schools website1. This includes words such as: return, arguments, if, try, var, setTimeout, catch, window, and length. Further explanations for the features extracted for creating a web application profile is described in detail in section 4.

The HTML code for the google search page is shown in Fig. 3. By classification purposes, this page using supervised machine learning and a complete page scan for features such as the keyword to word ratio or the number of functions might consider the page to be malicious because it contains many of the characteristics that have been shown to be typical of malicious scripting attacks.

```
[],yb=function(a){xb[0]=a},zb=function(a,b){var c=b||{};c._sn="pw";A(a,c)},Ab=
{signed:xb,elog:zb,base:"https://plusone.google.com/u/0",loadTime:(new Date).getTime()};p.pw=Ab;var
Bb=function(a,b){for(var c=b.split("."),e=function(){var m=arguments;a(function(){for(var
j=l,k=0,s=c.length-1;k<s;++k)j=j[c[k]];j[c[k]].apply(j,m)})},g=l,f=0,h=c.length-
1;f<h;++f)g=g[c[f]]=g[c[f]]||{};return
g[c[f]]=e};Bb($,"pw.clk");Bb($,"pw.hvr");n("su",yb,l.pw)};function Cb(){function a()
{if(document.getElementById("gb")==null)B(X.i,{_m:"nogb"});else{for(var
k;k=f[h++];)if(k[0]=="m"||k[1].auto)break;if(k)
{w(2,k[0]);v(k[1].url)}h<f.length&&setTimeout(a,0)}}function b(){g-- >0?setTimeout(b,0):a()}var
c=_tvv("1"),e=_tvv(""),g=3,f=q,h=0,m=window.gbarOnReady;if(m)try{m()}catch(j){A(j)}if(e)n("ldb",a);
if(c)window.addEventListener?window.addEventListener("load",b,i):window.attachEvent("onload",b);els
b()}
n("rdl",Cb);}catch(e){window.gbar&&gbar.logger&&gbar.logger.ml(e);}})();
(function(){try{var b=window.gbar,c=function(a,d){b[a]=function()
{if(window.navigator&&window.navigator.userAgent)return d(window.navigator.userAgent);return
false}},e=function(a){return!(/AppleWebKit\/.+
(?:Version\/[35]\.|Chrome\/[01]\.)/.test(a)||a.indexOf("Firefox/3.5.")!=-1)},f=function(a){return
a.indexOf("iPad")==-1};c("bs_w",e);c("bs_s",f);}catch(e){window.gbar&&gbar.logger&&gbar.logger.ml(e
();
(function(){try{var a=window.gbar;a.mcf("sf",{});}catch(e)
```

Figure 3. Portion of the HTML extracted from the google search page.

While dynamic systems have been shown to have a high success rate in correctly classifying true positive (TP) malicious pages, the analysis is time consuming [13, 16]. After successful identification of a page that has injected XSS, the virtual machine on the honeyclient will need to be restored since the platform cannot be trusted after an attack [16]. The time and resources required for this process are therefore expensive.

Current implementations of anomaly IDS using unsupervised clustering have typically been relegated to network-based systems.

We propose the creation of a host-based, anomaly detection IDS that can accurately classify TP pages with a low number of FN. Misclassification of FP web applications can also be achieved using our methodology. Contributions of the IDS we propose will contain the following:

- Analysis will be performed on the web application as a whole, thus deviating from same-origin policy.
- Domain based – all profiles created from extracted features will be kept in a database specific for that specific domain.
- The use of unsupervised clustering to detect anomalies in JavaScript and HTML content found in the web application.
- Proposed future studies to complete the classification of XSS in a web application by supervised learning or dynamic analysis of web pages found to be suspicious.

Like the Profiler, we will be scanning the whole page for features used to create the profiles of each web page before mentioned. This will eliminate the chance of mistaking web application code for client untrusted areas of code or in the case of missing malicious scripts, because code was not injected into normal client input areas.

IV. METHODOLOGY

The goal of our process is to separate benign web application pages from pages with added cross-site scripting (XSS) attacks with low number of false negatives (FN). This indicates that a page with an injected script is classified as benign. Anomaly-based detection systems typically have a higher number of false positives (FP) than misuse detection [16]. FP indicates a benign page that is categorized as malicious. Other static applications using supervised neural networks have been shown to classify true positive (TP) scripts within 94.4% accuracy for drive-by download applications [16, 20] and 92.0% accuracy for obfuscated JavaScript attacks [19] when the scripts themselves have been provided and the training data is up to date [17]. Therefore, after anomaly detection, we further propose a hybrid model combining the clustering of web application profiles by their anomalies and then extracting the scripts in question and using a supervised neural network for classification of the script. For our methodology, we will focus on the detection of anomalies with low FN assignments.

Application of our intrusion detection system (IDS) for anomaly cross-site scripting (XSS) attack detection is carried out in a 4-stage process: First – a web crawler is employed to extract all embedded HTML from the web application being

[1] http://www.w3schools.com/js/js_reserved.asp

examined. Second – the HTML is then parsed for different features for the purpose of creating the profile of the page. Third – the profile is compared with other profiles created in the database by use of a k-means algorithm. Fourth – the profile is either saved into the database with other profiles if it clusters among the other profiles, or it is quarantined and defined as an intrusion if it differs significantly from the other profiles or best matching unit (BMU).

A. Two Platforms for Testing

which to conduct experiments. Two different testing platforms are created. One is for controlled testing using profiles that are predicable. The second is a real world environment with previously deemed benign web pages.

For the controlled area of testing, a simple test page we write is created. The page contains simple HTML structures, an executable, and an area for user input which is run as a local host for preliminary testing. The code for this page is shown in Fig. 4. This page is replicated two more times, with the addition of an executable, and the addition of an HTML element, respectively, directly above the comments section. The comments section is used for client side inputs.

The second platform for testing, we take sites from DMOZ2 as our sample group. These websites come from a controlled source and are reputed to be benign and free of any injected malicious code.

The process for extracting the embedded HTML, parsing into features, and organizing the data from both testing platforms follow the same procedures.

Web applications to be analyzed are collected using a web crawler to extract HTML code embedded in the page. The open source web crawler we used is HTTrack3.

```
<'DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>XSS Test original</title>
    </head>
    <body>
        <h1>XSS Test Page with original content</h1>
        <h2>Controlled Environment </h2>
        <script type="text/javascript">
            document.write(<p> + Date() +"</p>");
        </script>
        <p>Javascript to be inserted below for a simulated persistent attack</p>
        <from method="post" action="XSSTest_original.jsp">
            <label for="content"> Post Comment </label>
            <textarea id="comment" name = "comment" id= "content"></textarea>
            <input type= "submit" value= "Submit"/>
        </from>
    <comment>
        <%
            String comment = "";
            // a string variable holds the path of the text file that stors all comments.
            string txtFilePath = "C:/Users/Ephraim_Nielson/Desktop/REU2016/XSSTest_originalText.txt";
            // get the text from the textarea and put in a string variable (comment)
            comment = request.getParameter("comment");
            //-----------------------------------------------
            // check the text file to print the comments when the page loaded
            //-----------------------------------------------
            txtFilePath = "C:/Users/Ephraim_Nielson/Desktop/REU2016/XSSTest_originalText.txt"
            BufferedReader reader = new BufferedReader(new FileReader(txtFilePath));
            StringBuilder sb = new StringBuilder();
            String line;
            // get all the comment from the text and append them in one string
            while ((line = reader.readLine()) != null) {
                sb.append(line + "</br>");
            }
            // print the all comments
            out.println(sb.toString());
            // check the variable comment if it has a text.
            // when the pge load at the begining, this condition will be evaluted false
            // because I have not submitted any thing in the textarea
            if((comment != null)) {
                //-----------------------------------------------
                //  add a comment to the text file.
                //-----------------------------------------------
                txtFilePath = "C:/Users/Ephraim_Nielson/Desktop/REU2016/XSSTest_originalText.txt";
                try {
                    PrintWriter pw = new PrintWriter(new FileOutputStream(txtFilePath, true));
                    pw.println(comment);
                    out.print(comment + "</br>");
                    //clean up
                    pw.close();
                }catch (IOException e) {
                    out.println(e.getMessage())
                }
            }
        %>
    </comment>

</body>
</html>
```

Figure 4. Test page code

B. Features

The features of the web application are our way to map the amount and characteristics of scripting and HTML content. These individual features we put into a vector called the feature vector, or the profile, of that page. This is indicative of the state the page is currently in and, when compared to past and future profiles, is indicative of the web application state whether its in a protected state or in an intrusion state. In the process of extracting features, no code is executed by design for static analysis. This will allow our IDS to analyze the page quickly [16].

Feature extraction is performed by parsing the HTML document we create using Java. The JSOUP library is used to import the HTML document as a string to analyze. We extract 11 features from that string that we will explain in more detail below. There are many other features that will give a more accurate profile of the web application state. The extraction of those features have been performed in other works and are recommended to be done for future work. The scope of our methodology we limit for testing purposes of our model so full extraction of features are not performed. These other features can be found in the published works by Canali et. al. [16], Likarish, Jung, and Jo [19] and Nunan et. al. [10]. The 11 features we use are as follows:

- Keyword to word ratio – these keywords we use can be found on the w3school website.
- Percentage of script content on the page – the percentage of content between <script></script> tags compared to the rest of the HTML content of the page.
- Percentage of whitespace content on the page.

[2] http://www.dmoz.org
[3] https://www.httrack.com

- Total embedded objects – a full description of these can be found on the w3school website4.
- Total URLs – amount of URL redirects found on the page.
- HTML elements with small areas – elements with less than 10 pixels per side.
- Total <script> tags
- Double documents – repeating references to documents or external resources.
- Functions used for de-obfuscation routines
- Total number of functions – variables assigned as a function().
- Total number of the HTML event handlers – setTimeout, setInterval, onclick, and onmouseover.

These features are then organized into a vector we use as a profile of our web application. The first one we use as our BMU since we assume it is benign. Subsequent profiles are made for the domain as interaction occurs and are mapped in relation to each other by means of clustering. We perform these actions using our two different platforms. First the controlled testing using profiles that are predicable. The second is a real world environment with previously deemed benign web pages.

C. *Organizing Testing Platforms for Analysis*

For the first platform, we take our three pages and inject benign HTML code in the form of comments. These are then web-crawled for embedded HTML content, parsed, and organized into profiles as described above. We also inject a page with a redirect to a malicious external resource4 and make its profile. Data collected from this domain and its results will be covered in section 5.

For the second platform, we similarly take the domain and its pages as the BMU of our clustering map. Several of the pages we inject with code for the Samy Worm that was used in the MySpace attack in 2005 [21].

This code is used as a more realistic attack in a real world setting because the attack used for our first platform has a signature typical of attacks, which is not allowed to be executed in current browsers. Results for this platform are discussed and illustrated in Section 5.

V. RESULTS.

In measuring the results of our two different platforms, it is important to remember that the goal of our methodology is to have the amount of false negative (FN) classification of profiles to be as low as possible. While false positive (FP) results are not as important for the reason that further testing will need to be applied for definite classification, our goal is to reduce the number of FP clustering as well. Overall accuracy will be an analysis of all the apparent clustering and the states of their profiles whether injected with malicious content or not.

The metrics we will use will be based on the clustering of the profiles. Benign profiles clustering around the best matching unit (BMU) or the assumed benign profile we will assign as true negative (TN). Malicious files that cluster around the BMU we will call false negative (FN). Benign profiles that deviate from the cluster we call a false positive (FP). Finally, the profiles deviating from the BMU cluster that are assigned to the web application with the injected attack we call true positive (TP). Overall accuracy (ACC) of the system can be shown by (1). True positive rate (TPR) or sensitivity of the system is shown in (2). True negative rate (TNR) or specificity is shown in (3).

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}. \quad (1)$$

$$TPR = \frac{TP}{TP+FN}. \quad (2)$$

$$TNR = \frac{TN}{TN+FP}. \quad (3)$$

Using these metrics, we can measure the accuracy and feasibility of our system for future work and implementation. The first experiments are carried out on the supervised support vector machine (SVM) to illustrate the clustering of such profiles according to their features. This is done for clarification and a visual representation of benign and suspicious profiles. In the other part of testing, we use unsupervised k-means clustering to find deviations for further analysis. Our tools we use for our SVM and k-means algorithms are found from the open source Orange design application5 from python.

A. *Results of Controlled Environment System*

The first environment to test we use the first platform we designed. We create 8 different benign profiles using the techniques we stated in Section 4. This takes the profiles of the original page, one with an added HTML element, and another with an added executable. The remaining 8 profiles are created by adding content through the comments section that can include the use of HTML tags and JavaScript code. 5 are made by inserting benign client side code and 3 are made by inserting the redirect to a malicious external resource5. In this preliminary test, the classification of the profiles is known to illustrate the clustering visual, identify the deviations, and check for accurate clustering. The clustering of these profiles using a support vector machine (SVM) are shown in Fig. 5. The blue markers represent profiles that we already know are benign. The red markers represent the profiles that are known to contain the malicious redirect.

[4] <script src=http://www.hackers.org/js/xss.js></script>
[5] http://www.orange.biolab.si

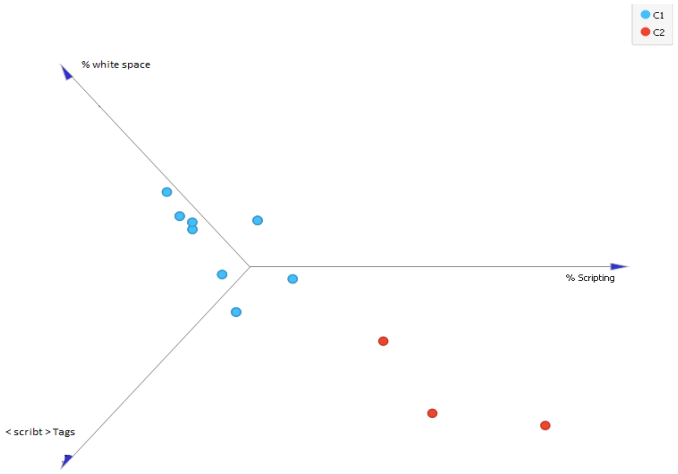

Figure 5. Illustration of clustering results using SVM against 3 axes labeled the percentage of whitespace, percentage of scripting content, and number of <script> tags.

For illustration purposes the results show a general clustering of the benign markers close to each other and the red group cluster away from the others. By using the SVM algorithm, we can separate the benign and malicious pages by the margin, as shown in Figure 5. The SVM probability distribution is shown in Fig. 6 to illustrate the probability of a certain profile being assigned benign or likely being assigned suspicious over an axis of the percentage of scripting to relative density. With the study's result, we want to investigate more in the suspicious webpages and apply more training on them. Our algorithm will continue processing the data until it repeatedly reach the lowest possible result.

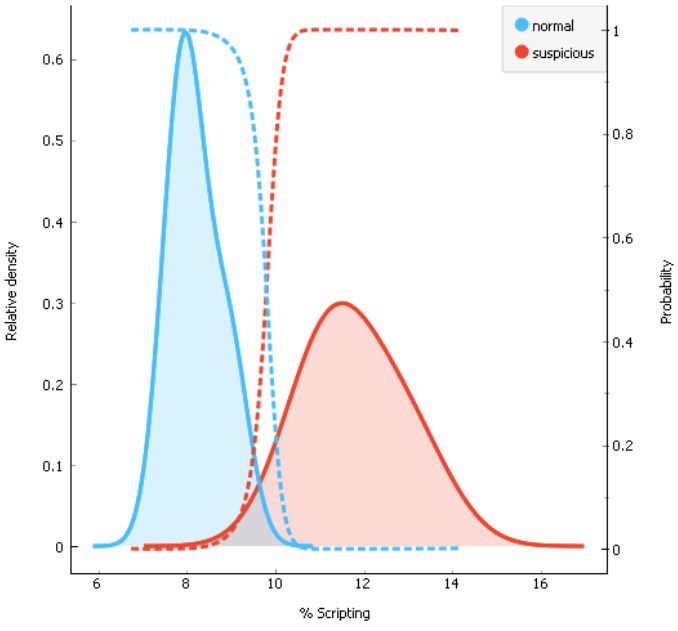

Figure 6. SVM probability distribution for profile organization and clustering

As is shown, the probability of a correctly clustering the 8 benign and 3 maliciously classified profiles correctly is more than 95% according to just the scripting content of the profiles. While there is possibility for FP and FN clustering, the combination of profile clustering according to the complete set of features is more accurate. An unsupervised clustering example is given in the next section to visually demonstrate the ability of such a network to cluster a number of profiles correctly. As will be shown, the more profiles added to the network, the more accurate the system clusters. This is important as previous work has shown to misclassify anomalies when attacks happen rarely [17].

B. *Result of DMOZ System*

In the second environment for testing, we took pages from the DMOZ domain. The HTML for each page was extracted using the HTTrack web-crawler we previously used. A simulated malicious injection attack was created by injecting a benign DMOZ page with the Samy worm [21]. While the Samy worm is now benign in browsers, the purpose of injecting this attack is to show a deviation from the current site style of design and formatting. Any anomaly and strong deviation from site design is expected to be indicated as suspicious.

A sample size of 270 profiles for testing are demonstrated for anomaly identification. This is done using a pre-labeled, known identification of each page and find clustering results. This is done using a various clustering algorithms. The second uses k-means for clustering and identifying suspicious clustering for further classification.

Using a SVM, we produced the results of the 270 profile sample in which 4 of the profiles were taken from pages injected with the Samy worm. The results shown in Fig. 7 show the probability of files being correctly clustered with a focus on scripting content.

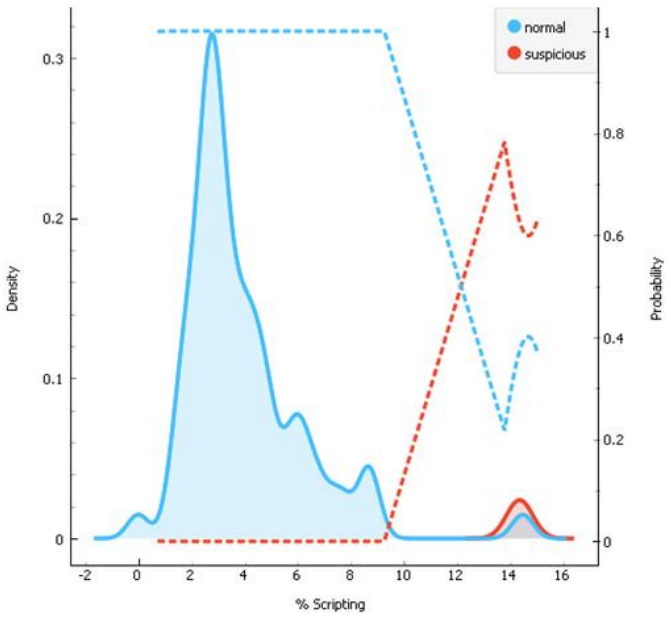

Figure 7. SVM probability distribution for organization and clustering for a sample size of 270 profiles with 4 injected profiles.

Clustering results according to scripting content show correct identification of the attacked sites or a TP of 4 and a FP rate of 1. The overall accuracy of the clustering is 99.6%, a TPR of 80% and a TNR of 99.6%. While some FP profiles were identified, the purpose of our work is to minimize the amount of FN which we found to not exist using the current method. Therefore, based off clustering, we can find the suspicious profiles and the web application pages they are associated with. From this we can see the feature of scripting being one of the prominent means of identifying attacks.

The same experiment is carried out using other means algorithms to determine the ability of a network to correctly cluster profiles according to full content. The algorithms we use are SVM, Logistic regression, Naïve Bayes, and k nearest neighbor. The results for these algorithms are shown in Table 1. As is shown, all suspicious profiles were correctly separated from normal profiles.

TABLE I. ACCURACY OF VARIOUS SUPERVISED CLUSTERING ALGORITHMS IN CORRECTLY CLASSIFYING A SAMPLE SET OF 270 PROFILES ACCORDING TO THEIR CONTENT

SVM	**Predicted**			
Actual		***Normal***	***Suspicious***	
	Normal	100.0%	0.0%	266 pages
	Suspicious	0.0%	100.0%	4 pages
		266 pages	4 pages	270 total

Logistic Reg.	**Predicted**			
Actual		***Normal***	***Suspicious***	
	Normal	100.0%	0.0%	266 pages
	Suspicious	0.0%	100.0%	4 pages
		266 pages	4 pages	270 total

Naïve Bayes	**Predicted**			
Actual		***Normal***	***Suspicious***	
	Normal	99.6%	0.4%	266 pages
	Suspicious	0.0%	100.0%	4 pages
		265 pages	5 pages	270 total

kNN	**Predicted**			
Actual		***Normal***	***Suspicious***	
	Normal	100.0%	0.0%	266 pages
	Suspicious	0.0%	100.0%	4 pages
		266 pages	4 pages	270 total

The second part of using this platform is to perform clustering using k-means to determine suspicious profiles based on deviation. Because clustering groups the webpages together based on certain behaviors, the malicious webpages can be discovered when they deviate from the benign webpages. The accuracy of clustering is shown on a map with different features being the various axis used. Clusters are visually represented with different colors and show how closely each profile relates to the other. Figure 8 shows the confidence of the clustering with respect to the BMU. Figure 9 shows the use of k-means clustering to organize the 270 profile sample set according to clustering with an emphasis on scripting features.

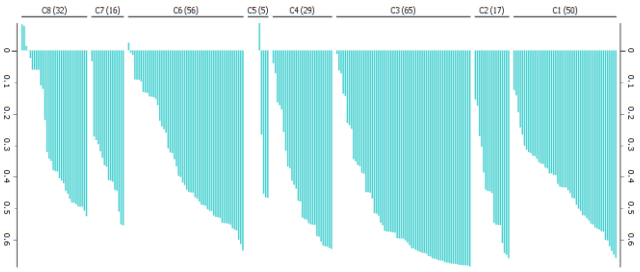

Figure 8. Confidence probability of a profile being correctly clustered according to the BMU.

Figure 9. Visual representation of clustering profiles according to 6 features in a sample of 50, 75, and 270 profiles left to right respectively.

It should be noted in Fig. 9 that specific group coloring of graph 1 does not necessarily correlate to the same grouping of profiles in graphs 2 or 3. In Fig. 10 we see that as the sample

size increases, the division between the main clustering of profiles increases. The third portion of the multi-dimensional scale (MDS) shows a distinct separation of the main clustering of profiles from the outliers. While definite classification is not available at this time of clustering, further work is proposed to classify the page by dynamic or data mining practices discussed in section 6.

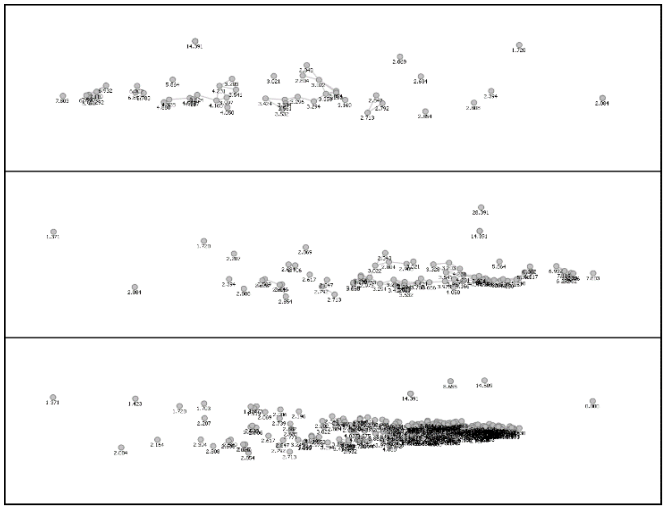

Figure 10. Multi-dimensional scale visual representation of clustering profiles according to the 11 features used in samples of 50, 75, and 270 profiles, top to bottom, respectively.

VI. Discussion

Web crawling and parsing occurs in the server, many times, cross-site scripting (XSS) attacks search for interpretation differences in each of the individual browsers and exploit those vulnerabilities [3]. This weakness has been the target of attacks. For example, the newline character '\n' used in the first large scale XSS attack in 2005 for the tag <java\nscript> bypassed sanitization processes but was executed as <javascript> in browsers that removed whitespaces [21]. While development of better Intrusion Detection Systems (IDS) that account for this type of attack is a continual process, the problem with interpreted languages – such as JavaScript – is that the assumption that server side parsing/rendering on the client side is consistent with the server side processing [3].

A proposed solution would be a periodic update of the features used in the vector to our model using data mining techniques after anomalous profiles have been classified as malicious. In this way, the web application acts as a honeypot and tracks the progress of attacks while gathering information on their methods. This will create better profiles of the JavaScript/HTML content of the web application for more accurate clustering as well.

VII. Conclusion

In this research work, we introduced a new method of a host-based, intrusion detection system (IDS) for scripting injection and persistent cross-site scripting (XSS) attack flagging. XSS is one of the top attacks used by hackers for identity theft, loss of assets, and information fraud. Current problems in identifying and correctly classifying such scripting attacks stem from a failure to correctly identify untrusted areas of injection and the lack of a signature for possible future attacks in the case of static analysis, and expensive processing requirements from dynamic analysis systems. Our methodology has shown to be accurate in clustering the state of a web application over a period of time and interaction using JavaScript and HTML snapshots of the web application, we call profiles. Deviant profiles place the web application in an intrusion state and are sent for further analysis that will need to be continued for future work.

VIII. Acknowledgements

This research work was conducted at Oakland University as part of the UnCoRe program - supported by the National Science Foundation under Grant No. CNS-1460897. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors would like to thank Dr. Huirong Fu, PI and Program Coordinator of the UnCoRe program at Oakland University.

References

[1] Fossil, M., Johnson, E., and Mack, T. Symantec global internet security threat report. Tech. rep., Symantec, 2016.Liang, Z., et al., The detection and quantification of retinopathy using digital angiograms. Medical Imaging, IEEE Transactions on, 1994. 13(4): p. 619-626.

[2] Percoco, N. J. Global security report 2010 analysis of investigations and penetration tests. Tech. rep., SpiderLabs, 2010.

[3] Nadji, Y., Saxena, P., Song, D. "Document Structure Integrity: A Robust Basis for Cross-Site Scripting Defense". In: 16th Annual Network & Distributed System Security Symposium, NDSS Symposium, 2009Heneghan, C., et al., Characterization of changes in blood vessel width and tortuosity in retinopathy of prematurity using image analysis. Medical image analysis, 2002. 6(4): p. 407-429.

[4] Wasserman, G. e Su, Z. "Static Detection of Cross-Site Scripting Vulnerabilities". In: 30th Internation Conference on Software Engineering, 2008.

[5] Uto, N., Melo, S.P. "Vulnerabilidades em Aplicações Web e Mecanismos de Proteção". Minicursos SBSeg 2009. IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Campinas, São Paulo, Brazil, 2009.

[6] Goldstein, M., Perlroth, N., & Sanger, D. E. (2014, October 03). Hackers' Attack Cracked 10 Financial Firms in Major Assault. Retrieved July 08, 2016, http://dealbook.nytimes.com/2014/10/03 /hackers-attack-cracked-10-banks-in-major-assault

[7] Yang, R., Kang, V., Albouq, S., & Zohdy, M. (2015). Application of Hybrid Machine Learning to Detect and Remove Malware. Transactions on Machine Learning and Artificial Intelligence, 3(4), 16.

[8] Langin, C., et. al., "A Self-Organizing Map and its Modeling for Discovering Malignant Network Traffic." (Mar 2009).

[9] Perlovsky and O. Shevchenko, "Cognitive neural network for cybersecurity," 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, 2014, pp. 4056-4061.

[10] Nunan, E., Souto, E., dos Santos, M., and Feitosa, E., "Automatic classification of cross-site scripting in web pages using document-based and URL-based features," Computers and Communications (ISCC), 2012 IEEE Symposium on, Cappadocia, 2012, pp. 702-707.

[11] Grossman, J., Hansen R., Petkov, D.P., Rager, A. e Fogie, S."Cross Site Scripting Attacks: XSS Exploits and Defense". Burlington, MA, EUA, Syngress Publishing Inc. 2007.

[12] Wichers, D., Dabirsiaghi, A., Di Paolo, S., Heiderich, M., Vela Nava, E. A., & Williams, J. (2013, October 29). Types of Cross-Site Scripting.

[13] Retrieved July 12, 2016, from https://www.owasp.org/ index.php/Types_of_Cross-Site_Scripting

[14] Ernst, M. "Static and dynamic analysis: synergy and duality". In Proceedings of WODA'2003 (ICSE Work-shop on Dynamic Analysis), Portland, pp. 25–28, May 2003.

[15] Xu, W., Zhang, F., Zhu, S., "JStill: mostly static detection of obfuscated malicious JavaScript code", Proceedings of the third ACM conference on Data and application security and privacy, February 18-20, 2013, San Antonio, Texas, USA

[16] Hwang, D. (2013). Cross-Site Scripting (XSS). Retrieved July 12, 2016, from http://hwang.cisdept.cpp.edu/swanew/Code.aspx?m=XSS

[17] Canali, D., Cove, M., Vigna, G., Kruegel, C., Profiler: a fast filter for the large-scale detection of malicious web pages. Proceedings of the 20th international conference on World wide web, Mar 2011, Hyderabad, India. ACM, pp.197-206, 2011,

[18] Wang, G., Hao, J., Ma, J., and Huang, L. "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering", Expert Systems with Applications: An International Journal, v.37 n.9, p.6225-6232, September, 2010

[19] Lee, S. C. and Heinbuch, D.V. "Training a neural-network based intrusion detector to recognize novel attacks," in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 31, no. 4, pp. 294-299, Jul 2001

[20] P. Likarish, E. Jung and I. Jo, "Obfuscated malicious javascript detection using classification techniques," Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on, Montreal, QC, 2009, pp. 47-54

[21] Konrad Rieck, Tammo Krueger, Andreas Dewald, Cujo: efficient detection and prevention of drive-by-download attacks, Proceedings of the 26th Annual Computer Security Applications Conference, December 06-10, 2010, Austin, Texas, USA

[22] Kamkar, S. (2005, October 5). MySpace Worm Explanation. Retrieved July 14, 2016, from http://samy.pl/popular/tech.html.

[23] Yang, R. R., Kang, V., Albouq, S., & Zohdy, M. A. (2015). Application of Hybrid Machine Learning to Detect and Remove Malware. Transactions on Machine Learning and Artificial Intelligence, 3(4), 16.

[24] Bazzi, T., Jasser, J., & Zohdy, M. (2016). Affine Arithmetic Self Organizing Map. matrix, 5(04). Chicago

[25] Millan, M. DeJonge, A. Alshammari, A. Alharbi, A. Zohdy, M. 2D HMM Application to IoT Security Attacks. IJICIC. In Progress.

Alshammari, A., Alhaidari, S., Alharbi, A., & Zohdy, M. (2017, June). Security Threats and Challenges in Cloud Computing. In Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on (pp. 46-51). IEEE.