Client-Based Web Attacks Detection Using Artificial Intelligence

Jiwon Hong
Korea Information Technology Research Institute
Hyeongmin Kim
Korea Information Technology Research Institute
Suhyeon Oh
Korea Information Technology Research Institute
Yerin Im
Korea Information Technology Research Institute
Hyeonseong Jeong
Korea Information Technology Research Institute
Hyunmin Kim
Financial Security Institute
Kyounggon Kim
kkim@nauss.edu.sa
Naif Arab University for Security Sciences

Research Article

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Javascript malware, Phishing attacks, Script-based web attacks

Posted Date: May 17th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2920883/v1

Additional Declarations: No competing interests reported.

Client-Based Web Attacks Detection Using Artificial Intelligence

Jiwon Hong[1†], Hyeongmin Kim[1†], Suhyeon Oh[1†], Yerin Im[1†], Hyeonseong Jeong[1†], Hyunmin Kim[2†] and Kyounggon Kim[3*†]

[1*]Best of the Best Program, KITRI, Seoul, 08378, Republic of Korea.
[2]Financial Security Institute, 132, Daeji-ro, Suji-gu, Yongin, 16881, Gyeonggi-do, Republic of Korea.
[3]Center of Excellence in Cybercrimes and Digital Forensics, Naif Arab University for Security Sciences, 11452, Riyadh, Kingdom of Saudi Arabia.

*Corresponding author(s). E-mail(s): kkim@nauss.edu.sa;
Contributing authors: hgzone323@gmail.com; oku00737@gmail.com; dhtngus20@gmail.com; dpfls9939@gmail.com; jung86357@gmail.com; hyunmini85@gmail.com.
†These authors contributed equally to this work.

Abstract

The prevalence of client-based web attacks, which exploit web vulnerabilities, has been increasing with the growth of web sites. Although pattern detection has been widely used to protect against web attacks, it has a high probability of failing to detect new types of attacks. To address this issue, we propose a novel approach for responding to three typical client-based web attacks (JavaScript malware, phishing attacks, and script-based web attacks) using machine learning algorithms. Our approach involves extracting relevant features from source code and URLs, and then training and testing various machine learning models (including Random Forest, Deep Neural Network, and Convolutional Neural Network) to determine the final model. Our experimental results indicate that our Random Forest model achieved high accuracy rates, with 99.99% for JavaScript malware, 95.11% for phishing attacks, and 94.77%

for script-based web attacks. Furthermore, we developed a Chrome extension that uses the learned models to block client-based web attacks.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Javascript malware, Phishing attacks, Script-based web attacks

1 Introduction

Due to the prevalence of online services offered by various organizations, websites have become a significant target for cyber-attacks [1]. As a result, extensive research has been conducted on attack methods and defenses against websites [2–5]. Attacks on websites can be classified into two categories: those targeting web server applications and those targeting website users. SQL injection is a common type of attack targeting web server applications, while cross-site scripting (XSS) is a script-based attack aimed at web users. Despite the increasing number of attacks targeting websites, over 70% of actual web attacks occur at the application layer [6]. In recent years, fileless cyber-attacks and Advanced Persistent Threat (APT) attacks have become prevalent [7–9]. These attacks infiltrate a company's internal network via a relatively easy-to-attack user's PC. Client-based web attacks such as phishing and script-based attacks are commonly used techniques to attack a user's PC.

Historically, defenses against client-based web attacks, such as JavaScript malware, phishing attacks, and script-based web attacks, have relied mainly on pattern detection techniques. Bo Sun's study, for instance, created a blacklist that automatically prevents access when the browser detects any pattern in the blacklist. The AutoBLG framework was also developed to automatically scan the web space and add new malicious patterns to the blacklist [10]. Nevertheless, attackers can easily evade these pattern detection techniques using code obfuscation or other evasion techniques [11]. Furthermore, there is a drawback to having all patterns exist in the database, as it can be resource-intensive and slow down throughput [12]. Therefore, we propose a method that utilizes various machine learning (ML) algorithms to detect client-side web attack techniques.

To detect client-based web attacks, our methodology proceeds as follows. First, we obtained a data set from the reputable and reliable Alexa Top Sites. We selected essential features using source code analysis and then applied RandomForest (RF), deep neural network (DNN), and convolutional neural network (CNN) models based on these features. We tested these three models and chose the one with the best accuracy as the final detection model. Lastly, we developed a Chrome extension that implements the validated model to safeguard real users against client-based web attacks.

The primary contributions of this research are outlined below: Firstly, we utilized three different ML algorithms for detecting client-side web attacks. Secondly, we gathered a significant amount of data by collecting 71,471 datasets for JavaScript Malware, 31,830 for Phishing, and 61,731 datasets for Script-based web attacks, which were the three main client-side attacks we focused on. Thirdly, we have developed a Chrome browser extension that utilizes the final RF model we selected.

The rest of this paper is structured as follows: Section 2 reviews the relevant literature on ML algorithms used to defend against client-side web attacks, Section 3 outlines the methodology used to collect and analyze datasets for detecting client-based web attacks, Section 4 describes the process of developing a model to detect three major client-based web attacks, namely JavaScript malware, phishing attacks, and script-based web attacks, Section 5 presents the experimental results of the final model, Section 6 describes the development of Chrome extensions based on the selected model, and Section 7 concludes the paper.

2 Related work

ML algorithms have been extensively studied for improving web application security. WANG Wei-Hong proposed a ML-based countermeasure against malicious JavaScript. In this study, 2,000 datasets were used to train three models (ADTree, NaiveBayes, and Support Vector Machine (SVM)), and the SVM model achieved the highest accuracy of 94.38% [13]. Hyunmin Kim et al. developed a web browser forensics toolkit that utilized ML algorithms. They collected a dataset of 52,500 web pages (10,000 benign and 42,500 malicious) and trained three models (SVM, DNN, and RF). Their research found that the RF model achieved the highest accuracy with 99.8% [14].

Monther Aldwairi conducted a study on detecting Drive-by-Download attacks, where models were trained on a dataset of 5,435 web pages and achieved an accuracy of 90% [15]. Arun Kulkarni proposed a ML-based approach for detecting phishing websites. They extracted nine features from a dataset of 1,353 URLs and developed four models (Decision Tree, Naïve Bayes' Classifier, SVM, and Neural Network). The Decision Tree model exhibited the highest accuracy of 91.5%, while the other three models achieved an accuracy of 80% [16].

S. Krishaveni and K. Sathiyakumari developed an XSS attack detection model that was trained on 500 URLs and produced a decision tree model with 100% accuracy [17]. However, most studies have used small datasets. Hyunsang Choi proposed a ML-based method for detecting client-based web attacks, such as spamming, phishing, and malware infection. This study collected 40,000 normal webpages and 32,000 malicious webpages and achieved 93% accuracy by applying the ML-kNN algorithm [18]. Joshua Saxe proposed a deep learning approach for detecting malicious webpages, and the architecture was designed for deployment on endpoints, firewalls, and web proxies [19]. Tom Diethe presented a pattern recognition system for detecting attacks on web services that target web servers and server-side applications [20].

There is a lack of research on detecting client-side web attacks that target users through their browser. Moreover, previous studies on JavaScript malware and phishing attacks have reported low model accuracy due to small dataset sizes and insufficient preprocessing procedures.

Previous research has mainly concentrated on identifying web attacks through ML algorithms on the server side. However, client-based defense methods can mitigate problems like overload and privacy concerns that may arise with server-side defense methods. A study by Jingyu Zhang presents a model for identifying cross-site request forgery (CSRF) attacks on clients by examining HTTP requests, content, and the CSRF handler in the browser [21]. Nonetheless, this method only identifies CSRF attacks.

This paper presents the development of three detection models for client-based web attacks (JavaScript malware, Phishing attacks, and Script-based web attacks) using over 30,000 datasets. We have introduced a reliable dataset collection method and preprocessing process to improve the model's reliability. Additionally, we have evaluated the accuracy of the models using various ML algorithms and selected the model with the highest accuracy.

3 Methodology: Dataset collection

The overall methodology of this study is presented in Figure 1. The data used for the ML algorithm was collected by segregating it into two categories: normal data and malicious data. The normal data was gathered from the Alexa Top site, which is a tool that provides information about website traffic rankings on the web [22]. However, since relying on the Alexa Top site alone to determine a site's safety is not sufficient, two additional measures were applied to ensure the safety of the selected sites. In the first step, URLs that do not start with https were excluded from the list. In the second step, we utilized Google's Safe Browsing to select URLs of secure websites, which detects unsafe websites and ensures the final dataset's safety.

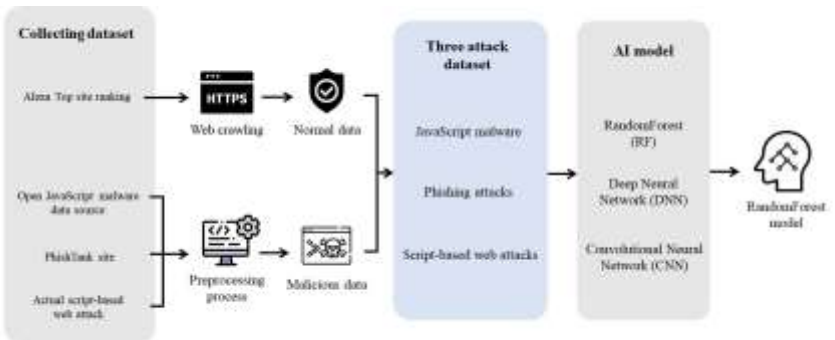


Fig. 1 Our Methodology: Collecting dataset, three attack dataset, and AI model structure.

The paper collects malicious data sets by different methods based on the attack types. Three types of malicious data are classified according to their characteristics, which include JavaScript malware, phishing attacks, and script-based web attacks. Each attack type has its own triggering location for malicious behavior. JavaScript malware injects a script into the browser via a vulnerability and executes the code as desired by the attacker. The dataset for JavaScript malware was obtained from an open malicious data source. Phishing attacks are attacks that aim to steal personal information through email and social network services, and the dataset for phishing attacks was collected from the phishing tank site. Finally, script-based web attacks, such as XSS and CSRF, are attacks that steal user privileges by accessing the user's cookies and session. The dataset for script-based web attacks utilized actual attack data sets provided by the industry.

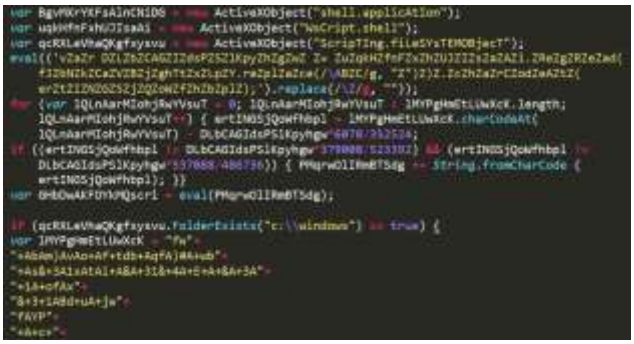
Table 1 presents a summary of the dataset that includes both normal and malicious websites, which are related to three types of client-side attacks. Subsequently, we will examine the specific code and preprocessing procedures for each of the malicious datasets.

Table 1 Dataset of normal data and malicious data

	JS malware	**Phishing**	**Script**
Data source	Javascript malware	Phishing Tank	script-based
Normal data	32,033	15,717	30,437
Malicious data	39,440	16,113	31,294

3.1 Javascript malware dataset

We analyzed the source code from URLs collected from open malicious data sources through web crawling and then extracted script tags from the source code. Figure 2 shows a sample of one of the JavaScript malware datasets.



```
var BgvMXrYKFsAlnCN1DS = new ActiveXObject("shell.applicAtIon");
var uqkHfmFxhUJIsaAi = new ActiveXObject("WsCript.shell");
var qcRXLeVhaQKgfxysvu = new ActiveXObject("ScripTIng.fileSYsTEMOBjecT");
eval(('vZaZr [illegible]'.replace(/\ABZC/g, "I")[illegible]
    [illegible]');').replace(/\Z/g, ""));
for (var lQLnAarMIohjRwYVsuT = 0; lQLnAarMIohjRwYVsuT < lMYPgHmEtLUwXcK.length;
    lQLnAarMIohjRwYVsuT++) { ertINGSjQoWfhbpl = lMYPgHmEtLUwXcK.charCodeAt(
    lQLnAarMIohjRwYVsuT) - DLbCAGIdsPSlKpyhgw[illegible];
if ((ertINGSjQoWfhbpl >= DLbCAGIdsPSlKpyhgw[illegible]) && (ertINGSjQoWfhbpl <=
    DLbCAGIdsPSlKpyhgw[illegible])) { PMqrwOlIRmBTSdg += String.fromCharCode (
    ertINGSjQoWfhbpl); }}
var GHbOwAKFDYkMQscrl = eval(PMqrwOlIRmBTSdg);

if (qcRXLeVhaQKgfxysvu.folderExists("c:\\windows") == true) {
var lMYPgHmEtLUwXcK = "fw"+
"+A&Am)AvAo+Af+tdb+AqfA)#A+ub"+
"+As&+3A1sAtAl+A&A+31&+4A+E+A+&Ar3A"+
"+1A+ofAx"+
"&+3+1A8d+uA+jw"+
"fAYP"+
"+A+c+"+
```

Fig. 2 Example of Javascript malware data

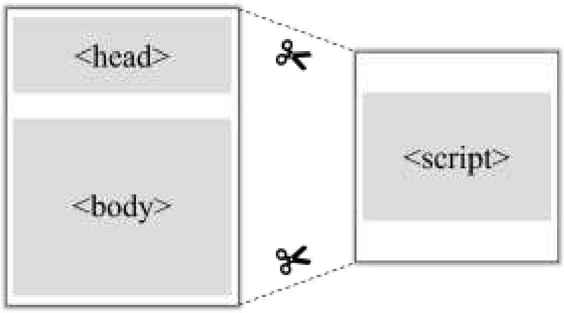
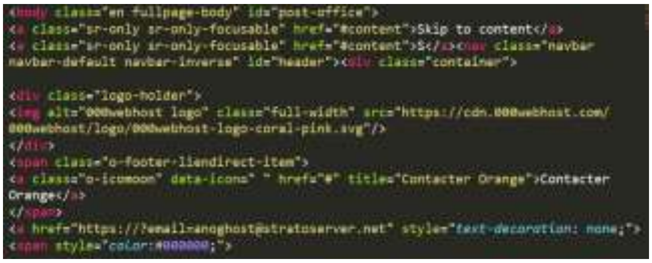


Fig. 3 Javascript malware data processing

The data processing method for JavaScript code is shown in Figure 3. We collected a total of 32,033 data from the URLs previously collected from the Alexa Top site as benign datasets. The malicious JavaScript dataset collected a total of 39,440 data from open JavaScript malware sites, as listed in Table 1 [23].

3.2 Phishing attacks dataset

We collected a total of 15,717 URLs and source codes as a benign site dataset for the phishing attack model. Figure 4 shows some of the phishing attack data. To detect phishing attacks more accurately, it is necessary to check not only the URL but also the source code of the website.



```
<body class="en fullpage-body" id="post-office">
<a class="sr-only sr-only-focusable" href="#content">Skip to content</a>
<a class="sr-only sr-only-focusable" href="#content">S</a><nav class="navbar
navbar-default navbar-inverse" id="header"><div class="container">

<div class="logo-holder">
<img alt="000webhost logo" class="full-width" src="https://cdn.000webhost.com/
000webhost/logo/000webhost-logo-coral-pink.svg"/>
</div>
<span class="o-footer-liendirect-item">
<a class="o-icomoon" data-icons=" " href="#" title="Contacter Orange">Contacter
Orange</a>
</span>
<a href="https://?email=anoghost@stratoserver.net" style="text-decoration: none;">
<span style="color:#000000;">
```

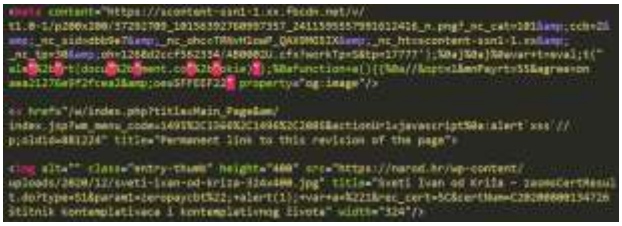
Fig. 4 Example of Phishing attacks data

We also collected a total of 16,113 phishing site URLs and source codes through Phishing Tank [24], which updates new phishing sites daily for malicious data.

3.3 Script-based web attacks dataset

We extracted not the entire source code of the website but tags containing frequently used attack patterns to extract the dataset of script-based web attacks. Figure 5 shows one of the script-based web attack datasets.

We extracted 7 tags: <a>, <input>, <img>, <script>, <meta>, <form>, and <div> from the source. The data processing method is shown in Figure 6.



```
<meta content="https://scontent-ssn1-1.xx.fbcdn.net/v/
t1.0-1/p200x200/37282709_10156392760997357_2411595557991012416_n.png?_nc_cat=101&amp;ccb=1&
amp;_nc_sid=dbb9e7&amp;_nc_ohc=TR0vM1owP_QAX0N0IIX&amp;_nc_ht=scontent-ssn1-1.xx&amp;
_nc_tp=30&amp;oh=1288d2ccf562334/A800002U.cfs?workTp=S&tp=17777');%0a}%0a}%0avar+t=eval;t("
ale%2Brt(docu%2Bment.co%2Bokie)");%0afunction=a(){[%0a//&optx1&mfayrt=55&agree=on
aea21276e9f2fcea1&amp;oe=5FFEEF22" property="og:image"/>

<a href="/w/index.php?title=Main_Page&am/
index.jsp?wm_menu_code=1495%2C1360%2C1496%2C2086&actionUrl=javascript%0a:alert`xss`//
p;oldid=881224" title="Permanent line to this revision of the page">

<img alt="" class="entry-thumb" height="400" src="https://narod.hr/wp-content/
uploads/2020/12/sveti-ivan-od-kriza-324x400.jpg" title="Sveti Ivan od Križa - zaomeCertResul
t.do?type=51&param1=zeropaycbt%22;+alert(1);+var+a=%221&rec_cert=5C&certNum=C2020000134726
štitnik kontemplativaca i kontemplativnog života" width="324"/>
```

Fig. 5 Example of Script-based web attacks data

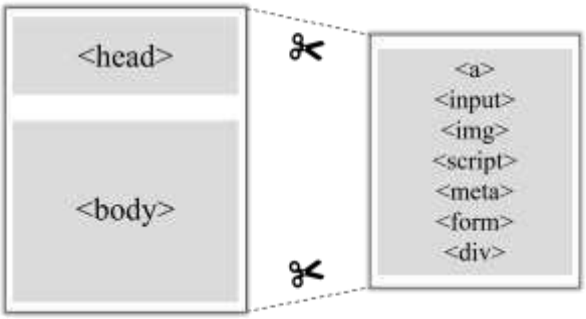


Fig. 6 Script-based web attacks data processing

The script-based malicious dataset created a new dataset by inserting cheat sheets into the seven extracted tags. From this insertion, we collected 30,437 benign data and 31,294 malicious data.

4 Model selection

Initially, we need to extract features for each attack to develop the model. After analyzing the collected dataset and each attack technique, we extracted the features and selected the final features through validation. For the feature validation task, we used a feature distribution visualization graph of the data to highlight the distribution of features briefly. The features that were finally selected through the feature verification process, accuracy, were derived through a total of three models: RF, DeepNeural Network (DNN), and CNN models.

4.1 Javascript malware

Through JavaScript malware analysis, strings and functions mainly used in browser exploits and drive-by attacks were selected as features. As a representative example, the split function and the join function, which are methods of recombining after dividing the string into pieces, were mainly used to avoid string-based detection. Additionally, the escape function that converts a string into ASCII code and the eval function that calculates and executes the JavaScript code were confirmed to appear frequently. These functions are used in malicious JavaScript for interpreting obfuscated sources.

Also, special characters used for obfuscation, such as exploit kits, were selected as features. Examples of special characters include \x, $, +, *, and /. Figure 7 shows the feature verification work for feature '\x,' and feature visualization was performed through the stripplot function.

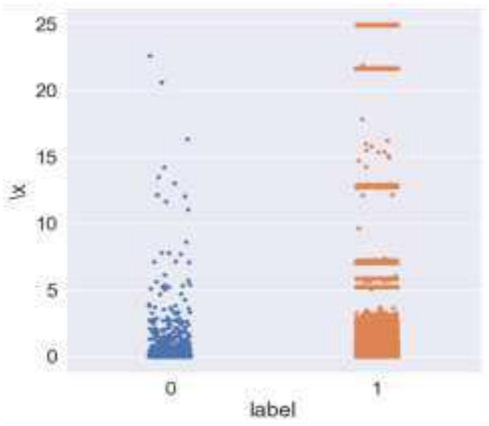


Fig. 7 '\x' feature validation graph

Label 0 refers to a benign dataset, label 1 refers to a malicious dataset, and it can be seen that the feature '\x' frequently appears at a higher rate in the malicious dataset. Therefore, it is considered a suitable feature for detecting JavaScript malware.

Figure 8 shows ADODB.Stream, which is commonly used in JavaScript malware, selected as a feature and verified. Through this, it can be confirmed that the feature 'ADODB.Stream' is a distinct feature used exclusively in malicious datasets.

Table 2 Javascript malware final feature list

\\x	0x	$	+
%	[	_	|
*	~	^	@
toString	Sleep	iframe	http://
onload	unonload	indexOf	charAt
WScript	Shell	Script	/.
eval(	escape(	join(	push(
split(	function	var	max_line
swf	exe	gif	display
ActiveXObject		ADODB.Stream	

After analyzing benign and malicious scripts, features were extracted, and finally, a total of 38 features were selected through feature verification, as shown in Table 2.

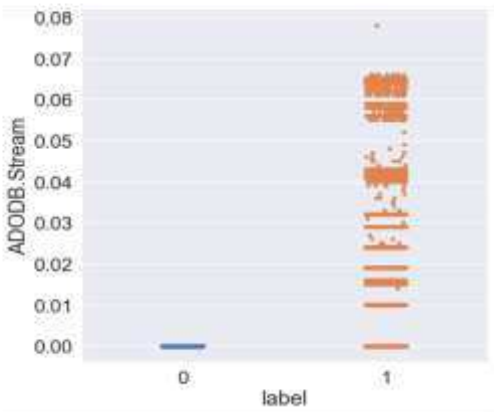


Fig. 8 'ADODB.Stream' feature validation graph

4.2 Phishing attacks

The features of phishing attacks were extracted by analyzing the URL and source code of the phishing site and comparing it with the benign dataset. As a result of identifying the features of the URL and the parts where malicious actions occur frequently in the source code, several features were discovered.

Figure 9 and Figure 10 display a portion of the collected benign URL dataset and the URL dataset of phishing sites. As a representative feature, it can be observed that phishing site URLs are, on average, longer than those of benign sites, and special characters appear frequently. Additionally, many cases were found where "//" was inserted in the URL path to redirect to another site, and due to the prevalence of obfuscated URLs, URL entropy was selected as a feature to determine obfuscation. By examining the domain registration period of phishing sites through Whois, it was confirmed that the majority of phishing sites were active for less than one year. Thus, URL-related features were extracted based on these characteristics.

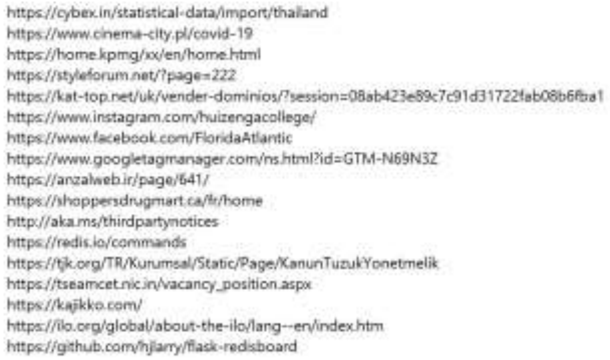
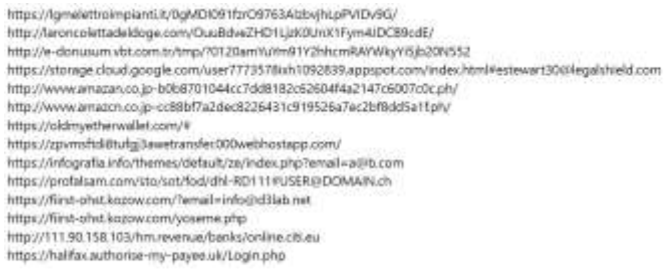

```
https://cybex.in/statistical-data/import/thailand
https://www.cinema-city.pl/covid-19
https://home.kpmg/xx/en/home.html
https://styleforum.net/?page=222
https://kat-top.net/uk/vender-dominios/?session=08ab423e89c7c91d31722fab08b6fba1
https://www.instagram.com/huizengacollege/
https://www.facebook.com/FloridaAtlantic
https://www.googletagmanager.com/ns.html?id=GTM-N69N3Z
https://anzalweb.ir/page/641/
https://shoppersdrugmart.ca/fr/home
http://aka.ms/thirdpartynotices
https://redis.io/commands
https://tjk.org/TR/Kurumsal/Static/Page/KanunTuzukYonetmelik
https://tseamcet.nic.in/vacancy_position.aspx
https://kajikko.com/
https://ilo.org/global/about-the-ilo/lang--en/index.htm
https://github.com/hjlarry/flask-redisboard
```

Fig. 9 Benign URL dataset of phishing site

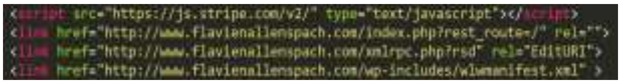


```
https://lgmelettroimpianti.it/0gMDI091fzrO9763AbbvjhLpPVIDv9G/
http://laroncolettadeldoge.com/OuuBdvaZHD1LjzK0UnX1Fym4IDC89cdE/
http://e-donusum.vbt.com.tr/tmp/?0120amYuYm91Y2hhcmRAYWkyYi5jb20N552
https://storage.cloud.google.com/user777357Bixh1092839.appspot.com/index.html#estewart300legalshield.com
http://www.amazan.co.jp-b0b8701044cc7dd8182c62604f4a2147c6007c0c.ph/
http://www.amazcn.co.jp-cc88bf7a2dec8226431c919526a7ec2bf8dd5a11.ph/
https://oldmyetherwallet.com/#
https://zpvmsftdi8tufg3awetransfer.000webhostapp.com/
https://infografia.info/themes/default/za/index.php?email=a@b.com
https://profalsam.com/sto/sot/fod/dhl-RD111#USER@DOMAIN.ch
https://first-ohst.kozow.com/?email=info@d3lab.net
https://first-ohst.kozow.com/yoseme.php
http://111.90.158.103/hm.revenue/banks/online.citi.eu
https://halifax.authorise-my-payee.uk/Login.php
```

Fig. 10 Malicious URL dataset of phishing site

The process of extracting source code-related features is as follows. In the source code of the phishing site, it was observed that external domains were entered in the href attribute value of the link tag. Additionally, the action attribute value of the form tag or website was disconnected and empty. Furthermore, there were numerous cases where external domains were present in the favicon path or script tag link.

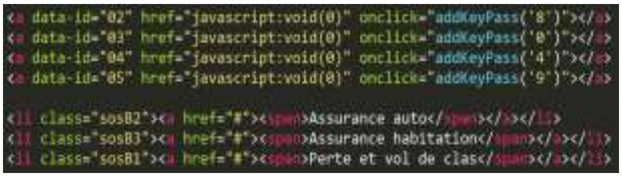
Figure 11 displays a portion of the source code from the phishing site 'http://it-friedli.ch/administrator/components/Login.htm'. Unlike the benign source code, it can be observed that an external domain is used as the href attribute value.



```
<script src="https://js.stripe.com/v2/" type="text/javascript"></script>
<link href="http://www.flavienallenspach.com/index.php?rest_route=/" rel="">
<link href="http://www.flavienallenspach.com/xmlrpc.php?rsd" rel="EditURI">
<link href="http://www.flavienallenspach.com/wp-includes/wlwmanifest.xml" >
```

Fig. 11 Example of malicious source code(external domain)

Additionally, there are many cases where the website is not linked through characters such as 'javascript:void(0)', '#', or 'about:blank' in the href attribute value, as shown in Fig. 12.



```
<a data-id="02" href="javascript:void(0)" onclick="addKeyPass('8')"></a>
<a data-id="03" href="javascript:void(0)" onclick="addKeyPass('0')"></a>
<a data-id="04" href="javascript:void(0)" onclick="addKeyPass('4')"></a>
<a data-id="05" href="javascript:void(0)" onclick="addKeyPass('9')"></a>

<li class="sosB2"><a href="#"><span>Assurance auto</span></a></li>
<li class="sosB3"><a href="#"><span>Assurance habitation</span></a></li>
<li class="sosB1"><a href="#"><span>Perte et vol de clas</span></a></li>
```

Fig. 12 Example of malicious source code (empty)

Features were selected through the analysis of URLs and source code, and based on this, feature verification was performed using feature importances. This can be seen in Figure 13. The graph indicates that the features 'https' and 'entropy' carry the most weight. As a result, a total of 23 features were extracted, as shown in Table 3.

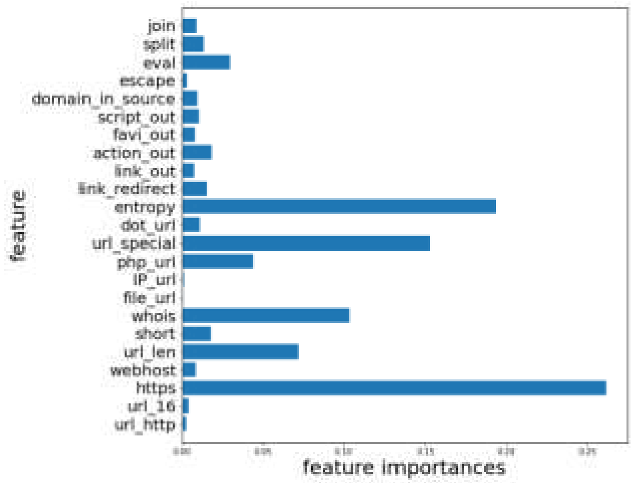


Fig. 13 Phishing attacks feature importances

Table 3 Phishing attacks final feature list

https	url_hex	url_len	url_short
url_special	url_dot	url_entropy	php_url
file_url	IP_url	webhost	whois
link_redirect	link_out	action_out	favi_out
script_out	domain_in_source		fromCharCode
split	join	escape	eval

4.3 Script-based web attacks

In the case of script-based web attacks, features can be classified into two categories. The first category consists of simple strings such as 'alert' and 'prompt', while the second category involves combinations of strings. In the combined form, tags and JavaScript strings with a high probability of attack coexist in the source code. Another case is when event handlers, alerts, 'document.cookie', etc., are used together. Additionally, when a tag or a single string is used, features were extracted for mixed case or an odd number of quotation marks.

The features extracted through the aforementioned process were verified using the feature importances of the RF model, as shown in Figure 14.

It can be observed that the features 'alerts' and 'xss' carry the most weight in script-based web attacks. Consequently, a total of 33 features were ultimately selected, as shown in Table 4, and the likelihood of encountering these features in benign data is very low.

5 Experimental Results

In this experiment, three algorithms, RF, DNN, and CNN, were used for learning. Firstly, the DNN constructed a neural network with two hidden layers of

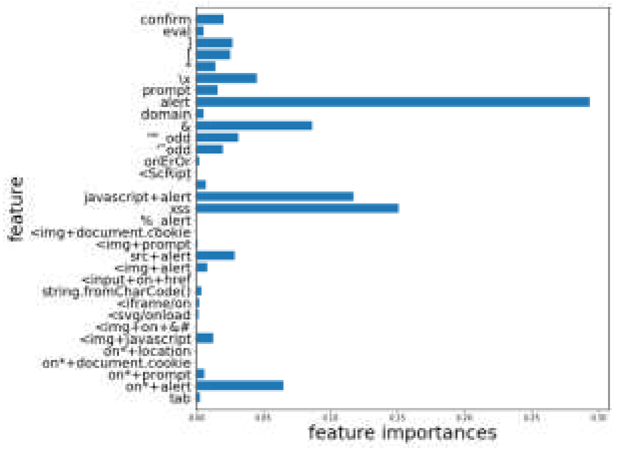


Fig. 14 Script-based web attacks feature importances

Table 4 Script-based web attacks final feature list

alert	prompt	'_odd	"_odd
onErrOr	<ScRipt	domain	xss
<ifame/on	<svg/onload	src+alert	%_alert
on*+alert	on*+prompt	on*+location	on*+cookie
<img+alert	<img+prompt	<img+cookie	<img+on+&#
<img+javascript		<input+on+href	
javascript+alert	string.fromCharCode()		eval
confirm	Tab	\x	&
*	'	[	]

size 14 and 7, and used a rectified linear unit (ReLU) as the activation function. Secondly, CNN constructed a neural network with two convolutional layers of size 64 and 32, and two hidden layers of size 32 and 16. ReLU was also used as an activation function. Additionally, a dropout layer was added between each layer to prevent overfitting in both DNN and CNN. For training, adam was used as the optimizer, and binary_crossentropy was used as the loss function. CNN and DNN were implemented using the Keras and TensorFlow libraries. Thirdly, RF was set to 50 trees and implemented using the scikit-learn library.

5.1 Model accuracy

Based on the previously collected data and features, three models were created for each attack: RF, DNN, and CNN. The first attack model, JavaScript malware, achieved high accuracy rates of 99.99%, 99.80%, and 99.83% for RF, DNN, and CNN, respectively, using 71,471 data samples and 38 features. The second attack model, Phishing attacks, yielded accuracy rates of 95.11% (RF), 93.71% (DNN), and 92.72% (CNN) with 31,830 data samples and 23 features. Lastly, the Script-based web attacks achieved accuracy rates of 94.77% (RF), 92.53% (DNN), and 92.38% (CNN) based on 61,731 data samples and 33 features.

Consequently, the JavaScript malware models consistently demonstrated high accuracy rates exceeding 99% across all three models, while the Phishing attacks model achieved 95.11% accuracy, with the RF model exhibiting the highest accuracy, as shown in Table 5. Similarly, the Script-based web attacks model also exhibited the highest accuracy with the RF model achieving 94.77% accuracy.

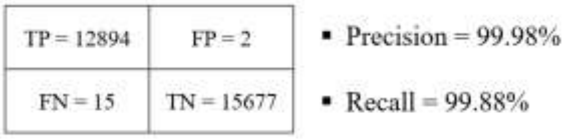
Table 5 Accuracy, Precision, and Recall of three attack type

	JS malware	**Phishing**	**Script**
Number of Data	71,471	31,830	61,731
Number of Features	38	23	33
RF Accuracy	99.99%	95.11%	94.77%
DNN Accuracy	99.80%	93.71%	92.53%
CNN Accuracy	99.83%	92.72%	92.38%
Precision	99.98%	98.51%	96.61%
Recall	99.88%	90.37%	88.06%

5.2 Model performance analysis

In all three models of JavaScript malware, phishing attacks, and Script-based web attacks, RF exhibited the highest accuracy. Based on the accuracy results of the three attack types, a performance analysis of the RF model for each attack was conducted using the confusion matrix.

The confusion matrix of JavaScript malware is depicted in Figure 15. It revealed a total of 2 false positives and 15 false negatives, indicating a precision of 99.98% and a recall of 99.88% for the model.

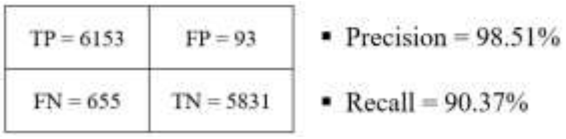


TP = 12894	FP = 2
FN = 15	TN = 15677

- Precision = 99.98%
- Recall = 99.88%

Fig. 15 JavaScript malware confusion matrix

The confusion matrix of Phishing attacks is depicted in Figure 16, revealing 655 false negatives and 93 false positives. Additionally, it can be observed that the precision of the model is 98.51% and the recall is 90.37%.

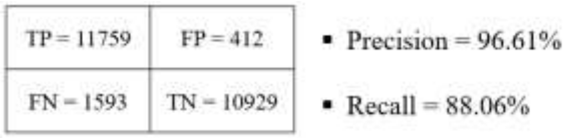


TP = 6153	FP = 93
FN = 655	TN = 5831

- Precision = 98.51%
- Recall = 90.37%

Fig. 16 Phishing attacks confusion matrix

In the last attack model, Script-based web attacks, a total of 1,593 false negatives and 412 false positives were identified. As depicted in Figure 17, the Precision and Recall of the model are 96.61% and 88.06%, respectively.



TP = 11759	FP = 412
FN = 1593	TN = 10929

- Precision = 96.61%
- Recall = 88.06%

Fig. 17 Script-based web attacks confusion matrix

6 Model application

We developed a 'Safe Browsing' Chrome extension that detects three attack models: JavaScript malware, phishing attacks, and script-based attacks, using our own data collection method, feature extraction, and verification. We utilized the RF model, which achieved a detection rate of over 94% for all three attack models, in the development of the Chrome extension [25].

The process of 'Safe Browsing' is illustrated in Figure 18. Since JavaScript malware, phishing attacks, and script-based web attacks are all client-side web attacks, 'Safe Browsing' operates on the client-side rather than the server-side.

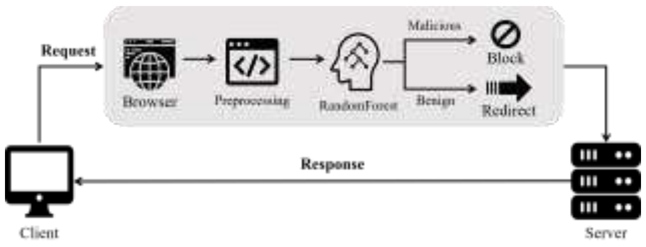


Fig. 18 'Safe Browsing' process

When a client sends a request to the server through a web browser, the requested website is checked for safety before the server sends a response. The URL and source code of the requested website undergo data preprocessing and are then input into the pre-existing RF model. Based on this input, the RF model determines whether an attack is detected. If a website has a malicious probability of over 90%, it is classified as malicious and blocks users' requests. Websites with a malicious probability below 90% are classified as normal and redirected to the server. The 'Safe Browsing' system's server periodically retrains the model to minimize false positives.

In addition to the percentage classification, the system offers the user an option to perform a safety test for each attack type and provides the test results. Among JavaScript malware, Phishing attacks, and Script-based web attacks, the user can selectively detect only the desired attack type through the safety check. By notifying the user about the safety check result, the system allows them to assess the malicious probability of the accessed website. Furthermore, the system includes a whitelist function that enables users to exempt specific websites from safety checks for the three attack types. Lastly, a concise explanation of JavaScript malware, Phishing attacks, and Script-based web attacks is presented to facilitate the understanding of feedback provided by the program, thus promoting awareness about the importance of website security among general users.

7 Conclusion

With the recent emergence of various bypass techniques for client-based web attacks, existing pattern detection methods have limitations in defending against these attacks. Furthermore, as attacks become more advanced, the damage range expands, starting from the client PC. In response to these evolving attacks, this study focuses on developing a client-based web attack countermeasure using ML algorithms.

This paper presents a detection model for JavaScript malicious code, phishing attacks, and script-based web attacks, which are frequently encountered during web attacks. The datasets used for training the models consist of 71,471 cases for JavaScript malware, 31,830 cases for phishing attacks, and 61,731 cases for script-based web attacks. Data preprocessing and feature extraction were performed for each attack type, followed by validation of the models using RF, DNN, and CNN algorithms. The RF algorithm achieved high accuracy rates of 99.99% for JavaScript malware, 95.11% for phishing attacks, and 94.77% for script-based web attacks. Based on this level of accuracy, we propose utilizing the RF algorithm as part of our approach.

To the best of our knowledge, this work represents the first study to apply a ML algorithm to counter client-based web attacks at the client end. In addition to calculating accuracy, we also consider precision and recall. Furthermore, we have developed a Chrome extension that directly applies the proven RF model in the browser to enhance user security.

Declarations

Ethical Approval. This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Authors' contributions. All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jiwon Hong, Hyeongmin Kim, Suhyeon Oh, Yerin Im, Hyeonseong Jeong, Hyunmin Kim and Kyounggon Kim. The first draft of the manuscript was written by Jiwon Hong and Kyounggon Kim, and all authors commented on previous versions of the manuscript. Review and editing were performed by Kyounggon Kim.

Funding. The authors did not receive funding from any organization for the submitted work.

Availability of data and materials. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Acknowledgements. This work was supported as part of the Next Generation Security Leader Training Program (Best of the Best) funded by Korea Information Technology Research Institute (KITRI).

References

[1] Kim, G.-H.: Implementation and design of proxy system for web vulnerability analysis. The Journal of the Korea institute of electronic communication sciences **9**(9), 1011–1018 (2014)

[2] Schütt, K., Kloft, M., Bikadorov, A., Rieck, K.: Early detection of malicious behavior in javascript code. In: Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence, pp. 15–24 (2012)

[3] Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D.C., Staples, J., Krause, L.: Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications **10**(1), 1–22 (2019)

[4] Rieck, K., Krueger, T., Dewald, A.: Cujo: efficient detection and prevention of drive-by-download attacks. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 31–39 (2010)

[5] Kim, H.Y., Kim, J.H., Oh, H.K., Lee, B.J., Mun, S.W., Shin, J.H., Kim, K.: Dapp: automatic detection and analysis of prototype pollution vulnerability in node. js modules. International Journal of Information Security **21**(1), 1–23 (2022)

[6] Yang, H.S., Yoo, S.J.: A study on secure model-based virtualization for web application security. Convergence Security Journal **14**(4), 27–32 (2014)

[7] Kim, K.-g.: State-Sponsored Hacker and Changes in hacking techniques. NetSec-KR Seoul, Korea (2017)

[8] Kim, K., Alfouzan, F.A., Kim, H.: Cyber-attack scoring model based on the offensive cybersecurity framework. Applied Sciences **11**(16), 7738 (2021)

[9] Lee, G., Shim, S., Cho, B., Kim, T., Kim, K.: Fileless cyberattacks: Analysis and classification. ETRI Journal **43**(2), 332–343 (2021)

[10] Sun, B., Akiyama, M., Yagi, T., Hatada, M., Mori, T.: Autoblg: Automatic url blacklist generator using search space expansion and filters. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 625–631 (2015). IEEE

[11] Choi, Y., Kim, T., Choi, S., Lee, C.: Automatic detection for javascript obfuscation attacks in web pages through string pattern analysis. In: International Conference on Future Generation Information Technology, pp. 160–172 (2009). Springer

[12] Uddin, M., Rahman, A.A.: Dynamic multi- layer signature based intrusion detection system using mobile agents. arXiv preprint arXiv:1010.5036 (2010)

[13] Wei-Hong, W., Yin-Jun, L., Hui-Bing, C., Zhao-Lin, F.: A static malicious javascript detection using svm. In: Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering, pp. 214–217 (2013). Atlantis Press

[14] Kim, H., Kim, I., Kim, K.: Aibft: Artificial intelligence browser forensic toolkit. Forensic Science International: Digital Investigation **36**, 301091 (2021)

[15] Aldwairi, M., Hasan, M., Balbahaith, Z.: Detection of drive-by download attacks using machine learning approach. In: Cognitive Analytics: Concepts, Methodologies, Tools, and Applications, pp. 1598–1611. IGI Global, ??? (2020)

[16] Kulkarni, A.D., Brown III, L.L., et al.: Phishing websites detection using machine learning (2019)

[17] Krishnaveni, S., Sathiyakumari, K.: Multiclass classification of xss web page attack using machine learning techniques. International Journal of Computer Applications **74**(12), 36–40 (2013)

[18] Choi, H., Zhu, B.B., Lee, H.: Detecting malicious web links and identifying their attack types. WebApps **11**(11), 218 (2011)

[19] Saxe, J., Harang, R., Wild, C., Sanders, H.: A deep learning approach to fast, format-agnostic detection of malicious web content. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 8–14 (2018). IEEE

[20] Corona, I., Giacinto, G.: Detection of server-side web attacks. In: Proceedings of the First Workshop on Applications of Pattern Analysis, pp.

160–166 (2010). PMLR

[21] Zhang, J., Hu, H., Huo, S.: A browser-based cross site request forgery detection model. In: Journal of Physics: Conference Series, vol. 1738, p. 012073 (2021). IOP Publishing

[22] Amazon: “Alexa the Top 500 Sites on the Web.” Accessed Nov. 26, 2020. https://www.alexa.com/topsites

[23] HynekPetrak github. Accessed Oct. 27, 2020. https://github.com/HynekPetrak/javascript-malware-collection

[24] OpenDNS: “anti-phishing Site.” Accessed Jan. 27, 2021. https://www.phishtank.com/

[25] oku00737: Safe Browsing. Seoul, Republic of Korea (2020. [Online].). https://chrome.google.com/webstore/detail/safe-browsing/nlabjhdjaeiajfgkpafhmdhnfeckpeol