Journal of Information Systems Engineering and Management
2025, 10(9s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Leveraging Signature Patterns and Machine Learning for Detecting HTTP Header Manipulation Attacks

Arti Deshpande[1], Bhushan Jadhav[2], Trisha Nadar[3]
[1]*Associate Professor, Department of Computer Engineering, Thadomal Shahani Engineering College, Mumbai, India.*
arti.deshpande@thadomal.org
[2]*Assistant Professor, Department of Artificial Intelligence and Data Science, Thadomal Shahani Engineering College, Mumbai, India.*
bhushan.jadhav@thadomal.org
[3]*Department of Computer Engineering, Thadomal Shahani Engineering College, Mumbai, India.*
trisha.nadar2604@gmail.com

ARTICLE INFO

Received: 05 Nov 2024

Revised: 28 Dec 2024

Accepted: 08 Jan 2025

ABSTRACT

Hypertext Transfer Protocol (HTTP) injection is a security vulnerability in which attackers manipulate HTTP Headers for malicious intent which facilitate various types of attacks like Downgrade-attack, Session fixation, Session hijacking, Cross-site scripting (XSS), Script injection, Referer forgery, Host header injection and Cache poisoning. These HTTP header manipulations can also be used for phishing and malware attacks. This study proposes leveraging signature attack patterns enhanced with Machine Learning (ML) and Deep Learning (DL) for detection of malicious header. HTTP request headers will be intercepted using Mitmproxy, and known attacks such as Downgrade attacks, Session fixation, Session hijacking, Token manipulation, Script injection will be detected based on their unique signatures. Malicious Internet Protocol (IP) addresses in the headers are detected using a blacklist sourced from the IPsum GitHub repository. Additionally, the malicious classifier model utilizes a hybrid approach for feature extraction based on Natural Language Processing (NLP) and traditional methods followed by generation of adversarial samples using Generative Adversarial Network (GAN). Multiple supervised ML and DL models are employed to classify URLs as phishing, malware, or benign and detect the specific attack type such as Referer forgery, Host header injection and other malware-related activities. The dataset is sourced from trusted repositories like Phishing URL dataset by Mendeley, Malicious URLs dataset from Kaggle and IPSum GitHub repository to construct a curated dataset. Adversarial samples generated using GAN are augmented in the dataset used for training the model to make it resistant to adversarial attack. The detection of Malicious HTTP headers using the proposed model is evaluated using performance metrics.

Keywords: HTTP Header Manipulation, HTTP Injection, Signature-Based Detection, URL Classification, Generative Adversarial Network, Deep Learning

I. INTRODUCTION

Hypertext Transfer Protocol (HTTP) is a protocol used for transferring messages between the web server and web client(browsers) over the internet. The web server hosts data resources which can be accessed by users using web browsers via HTTP request messages and HTTP response messages with the requested resource in its body. The structure of the HTTP request message is divided into request line, headers in key-value pairs containing additional information and an optional body. Similarly, a status line, headers and an optional body constitute the HTTP response message. An HTTP header consists of its case-insensitive name followed by a colon (:), then by its value [14]. The HTTP header injection vulnerability is a web application security term that refers to a situation when the attacker tricks the web application into inserting extra HTTP headers into legitimate HTTP responses [16]. This is used to carry out a variety of attacks, including HTTP response splitting, often referred to as Carriage Return Line Feed (CRLF) injection, Information disclosure, Cache poisoning, and Security bypass. The following attack types are examined in this study on malicious HTTP request headers:

(i)Downgrade-attack: The attacker forces the web applications to use less secure protocols like downgrading from Hypertext Transfer Protocol Secure (HTTPS) to HTTP which allows Man In The middle (MITM) attacks where the

messages exchanged between the client and server are intercepted and altered by the attacker [4]. This can be achieved by changing the value of Upgrade-Insecure-Requests header to 0.

(ii) Token manipulation: This attack involves the use of forged, stolen, or duplicate authorization or session tokens to impersonate a user, bypass authentication, or hijack sessions. The Authorization header contains the bearer token used for user authentication, while the Cookie header contains session tokens that help maintain the user's session. These tokens can be manipulated to perform unauthorized actions.

(iii)Session fixation: A session ID known to the attacker is enforced for use by the user, and then this compromised Session ID is used to hijack a session after the user logs in. Websites that do not change the session ID after login are particularly prone to these attacks. This is achieved by injecting a known session ID into the Cookie header before the user logs in and then exploiting it to take control of the user session after the user logs in.

(iv)Session hijacking: The attacker captures or steals a valid user session ID to mimic the user and breach security restrictions to achieve unauthorized access to sensitive user information. Similar to attack (ii), the Cookie header can be exploited in this attack.

(v) Script injection: The attacker injects malicious scripts through HTTP request headers or other user inputs. These scripts are often executed without the user's consent intended at manipulating the browser behavior. They are of various types like SQL injection and Cross-site scripting (XSS). Database queries are executed to manipulate database or access data without authorization in SQL injection.

(vi) Cross-site Scripting (XSS): Reflected XSS is a technique to exploit security vulnerabilities by permitting the attackers to inject malicious scripts which will be immediately executed by the browser allowing the attacker to obtain private user information [15]. This can be done by injecting JavaScript statements similar to (iv) in headers like Referer, User-agent, Host, Origin, Content-type, X-forwarded-for, X-request-id. XSS is a subset of script injection and the key difference between them is that while script injection targets the server-side application by injecting malicious code into the server, XSS targets the users of the server application by executing scripts in their browsers to affect them directly.

(vii) Host header injection: An attacker can spoof the Host header in the HTTP request with a malicious URL or IP address to give the request an appearance that it is coming from a safe domain to perform a range of attacks aimed at bypassing security controls and gaining unauthorized access [23].

(viii) Referer forgery: Referer header usually contains an URL or IP address of the web page that has sent the request to the server. By manipulating this header, the server can be deceived to assume that the request is coming from a trusted source.

This study proposes a dual approach to detect above mentioned attacks, which involves analyzing the headers to identify vulnerabilities using signature patterns of certain attacks and employing a deep learning model to detect malicious URLs within the Host and Referer. This enables the detection of specific attacks and their classification as benign, phishing, or malware.

II. RELATED WORK

Neda Ali [1] defines HTTP response header injection as the exploitation of security vulnerabilities by injecting content malicious in intent into the response headers sent by the web server. The attacks include Security bypass, Cache poisoning, Cross-site scripting (XSS), Session hijacking, and phishing.

Using supervised machine learning techniques, Ashley Laughter et al. [9] gathered web traffic data and examined the HTTP headers to identify malicious and benign requests. After the experimental research, the author concluded four important observations as

- The header usage of malicious and benign requests differs,
- Content-type, Accept-encoding and Accept-language are HTTP headers used to efficiently classify a request as malicious or benign with 93.6% accuracy,
- The malicious and benign request lines differ in their lengths and can be used to differentiate the HTTP requests with 96.9% accuracy.

W. Tao et al. [24] and L. Xu et al. [28] worked on eleven previously recognized features. J. McGahagan et al. [12] worked on 22 features after adding newly detected 11 features. The author found that the top two features, content-length and content encoding gzip, are prevalent from earlier study after employing ensemble approaches (RandomForest, AdaBoost, ExtraTree, and Gradient Boosting) to understand feature relevance. However, according to the authors, their method was able to identify the third feature, Transfer-encoding chunked. Vary accept header ranked fifth, third, and fourth while not sampling, over-sampling, and under-sampling respectively. Other recently discovered header fields of significance are X-XSS-protection, HSTS, and the X content-type header with value of nosniff. The average Matthews Correlation Coefficient (MCC) for the selected 22 features was better than for the 11 previously studied features.

Mizuno et al. [13] offer a novel approach to automating the feature extraction process from the HTTP headers by using an automatic template creation methodology based on the DBSCAN algorithm. By eliminating the less important traits and building the templates statistically without requiring any prior domain knowledge, this increases the detector's resilience. The suggested method by the author shows false positive rate below 1% while discriminating between hostile and benign traffic with up to 97.1% precision. The employment of DNN with four layers and adaptive moment estimation to optimize the classifier produced the maximum accuracy while SVM produced the second highest accuracy. The system does, however, draw attention to how ineffective it is in capturing HTTPS and UDP-based protocols.

Martin Grill et al. [6] use HTTP User-Agent Discrepancy Identification to detect malware. According to them, a user-agent field could look like one of these: Browsers used by legitimate users, empty, specific, spoof, and inconsistent. The study suggests methods for identifying malware that fits into one of three categories: Discrepant, Specific, or Empty. They model domain usage for empty User-agents and categorize sites visited by a minority as anomalous. The frequency of User-Agents among network users is used to classify unknown, non-browser User-Agents. The authors note that a single user only utilizes a single web browser version on a single computer. This data serves as the foundation for identifying unusual, well-known browser User-Agents. The authors point out that a single user only uses one version of the web browser on one computer. Identification of uncommon, well-known browser User-Agents is based on this data. It is verified whether the user updated their browser if the User-Agent is different from an older one. If not, it is classified as abnormal. However, User-Agent feature cannot be identified to detect malware that uses Spoofed User-Agent.

According to Reyes-Dorta et al. [18], the false negatives of the confusion matrix should receive particular attention when comparing models in this area of cybersecurity since they indicate that harmful URLs are being taken into account as legitimate. The F1-score is therefore the optimum metric for comparing models, according to this study. This study suggests three distinct neural networks and concludes that using the "relu" activation function followed by the Sigmoid activation function in the output layer produced the best results. Additionally, the "adam" optimizer, the "binary_accuracy" metric, and the "binary_crossentropy" loss function were employed.

Comparison of various ML algorithms used in previous research studies is given in table 1.

Table 1: Comparison of methodologies used in previous research studies

Year	Author name	Proposed method	Accuracy given	Dataset used	Description
2023	Abdul Karim et al. [8]	Ensemble model based on SVM, Linear regression, and decision tree	95.23%	Kaggle: Phishing website detector by Eswar Chand	The feature selection technique based on canopy method coupled with cross fold validation and grid search hyper parameter tuning technique for phishing detection using malicious URLs
2023	Sanjeev Shukla et al. [20]	Random Forest algorithm	97.8%	Not mentioned	16 new HTTP headers, primarily security headers were used to determine if the web page was phishing or legitimate

2022	Tiefeng et al. [25]	Bidirectional Gated Recurrent Unit (DA-BiGRU)	97.92%	Kaggle: Malicious URLs dataset by Manu Siddhartha	Word2Vec is used to train word vectors and attention mechanism is introduced to learn sequence correlations enhancing the malicious URL detection
2021	Zhiqiang Wang et al. [27]	Dynamic convolutional neural network (DCNN)	98.7%	Malicious URLs: GitHub, uci.edu, Kaggle Benign URLs: Alexa	The pooling layer is replaced with k-max pooling, and a new folding layer is added for malicious URL detection. The pooling parameters are dynamically adjusted based on the URL length and current layer depth.
2021	Ashley Laughter et al. [9]	Supervised machine learning algorithm	93.6%- 96.9%	Not mentioned	Detects malicious HTTP request using headers and line length of HTTP headers
2018	Mizuno et al. [13]	Deep neural network (DNN)	97.1%	Malwr, TrendMicro, Kaspersky,MalShare, VirusShare, Campus network filtered using MalwareDomain Blocklist	Four layered DNN with adaptive moment estimation and automatic template generation using DBSCAN for malware detection
2018	Buber et al. [3]	Random Forest algorithm	97.2%	Malicious URLs: PhishTank and Yandex Search API	The algorithm utilized NLP-based features combined with vectorization-based features for detection of phishing URLs
2016	Vanhoenshoven et al. [26]	Multilayer perceptron (MLP)	97.28%	Dataset provided by Ma et al. [10]	A feature set comprising the features with the highest absolute Pearson coefficients relative to the prediction class is used in this feed-forward artificial neural network model

Rasheed et al. [17] states that adversarial attacks exploit security vulnerabilities in ML & DL by making minimal modifications to the malicious URLs using greedy approach which will lead to its misclassification as benign URL by the model. This research uses the Blackbox scoring strategies-DeepWordBug algorithm proposed by J. Gao et al. [5] to identify key segments or characters of the URL that, when altered, can lead to misclassification. The authors test their attack against three kinds of CNN based classifiers. An accuracy decrement of 60% for Character-based CNN model, 77% decrease for word-level CNN model and 56% decrease for a joint CNN model was observed. Augmenting the adversarial samples in the training set and adding domain name of detected malicious URLs to a blacklist is suggested as a way to make the detection model robust against these attacks.

III. PROPOSED METHOD

The initial GET request forwarded to the web server when the user initially searches for the URL is intercepted using Mitmproxy. Mitmproxy is a proxy server that allows to intercept and record all HTTP & HTTPS communication. The intercepted headers are displayed to the user, allowing them to modify the header values. The

proposed model analyzes the modified headers and detects any malicious attempts using a dual approach. The flow of the proposed model is illustrated in figure 1.

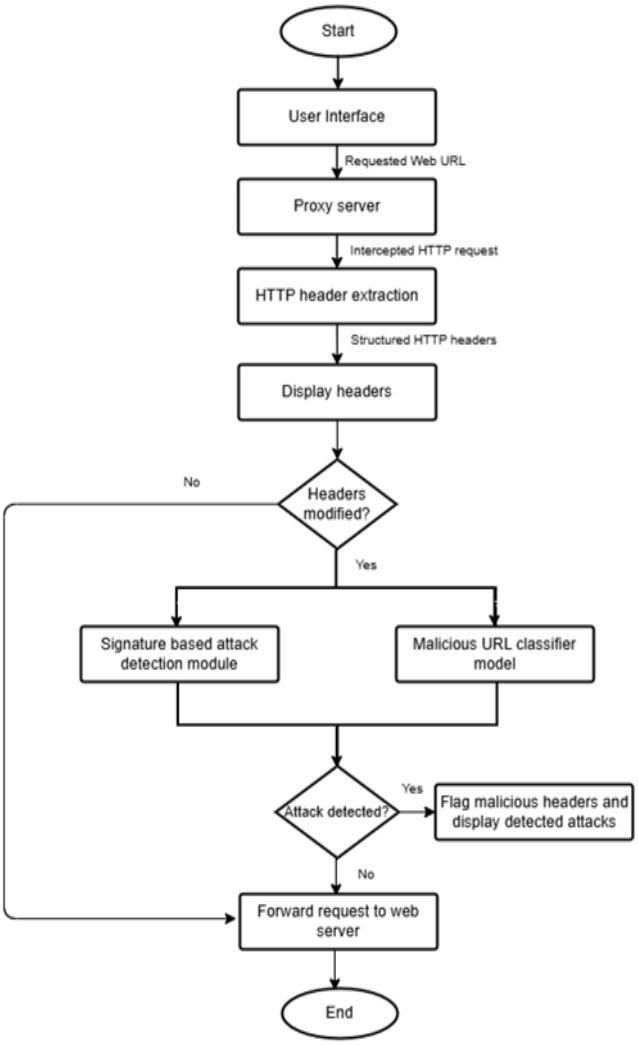

Figure 1: Flow diagram of proposed method

The signature-based attack detection module extracts the following headers if modified and analyzes their values to detect signature-based attacks:

(i)Upgrade-Insecure-Requests: A value of 0 in this header indicates a potential downgrade attack.

(ii)Authorization: This header is used to detect token manipulation attacks by performing basic validation of the token's structure and format. The standard JSON Web Tokens (JWT) format in the Authorization header consists of three base64url-encoded segments separated by periods and preceded by the Bearer keyword (Bearer <token>) [2]. For advanced token validation, the JWT secret key and the signing algorithm used by the web server are required, but these are not accessible as they are private to the server-side application. Therefore, the scope of this detection is limited to identifying violations of the JWT structure.

(iii)Cookie: If the session ID is dynamically assigned by the user in the Cookie header, it indicates a potential session fixation attack. Session hijacking, on the other hand, involves the use of a stolen valid session ID. Detection of Session hijacking therefore requires access to server-side session data which is not accessible to the client. Therefore, modification of Cookie header with session id is flagged as potential Session fixation or hijacking attempt.

(iv) X-Forwarded-For, X-Request-Id, Referer, User-Agent, Host, Origin, Content-Type: If SQL query patterns are detected within these headers, the request is flagged as a potential SQL injection attack. Similarly, if JavaScript keywords are found within <script> tags, the request is flagged as a potential Cross-Site Scripting (XSS) attack.

(v) Host: The URL or IP address present in the Host header is checked for host header injection. The IP addresses are validated for malicious activity using a blacklist sourced from the IPsum GitHub repository [21]. URLs are forwarded to the Malicious Classifier model, which is the second step in our dual detection approach.

(vi) Referer: The detection of Referer forgery follows the same step as given in (v).

After detecting these attacks and flagging the exploited headers, URLs in the Host and Referer headers are classified as phishing, malware or benign using the malicious URL classifier model illustrated in figure 2.

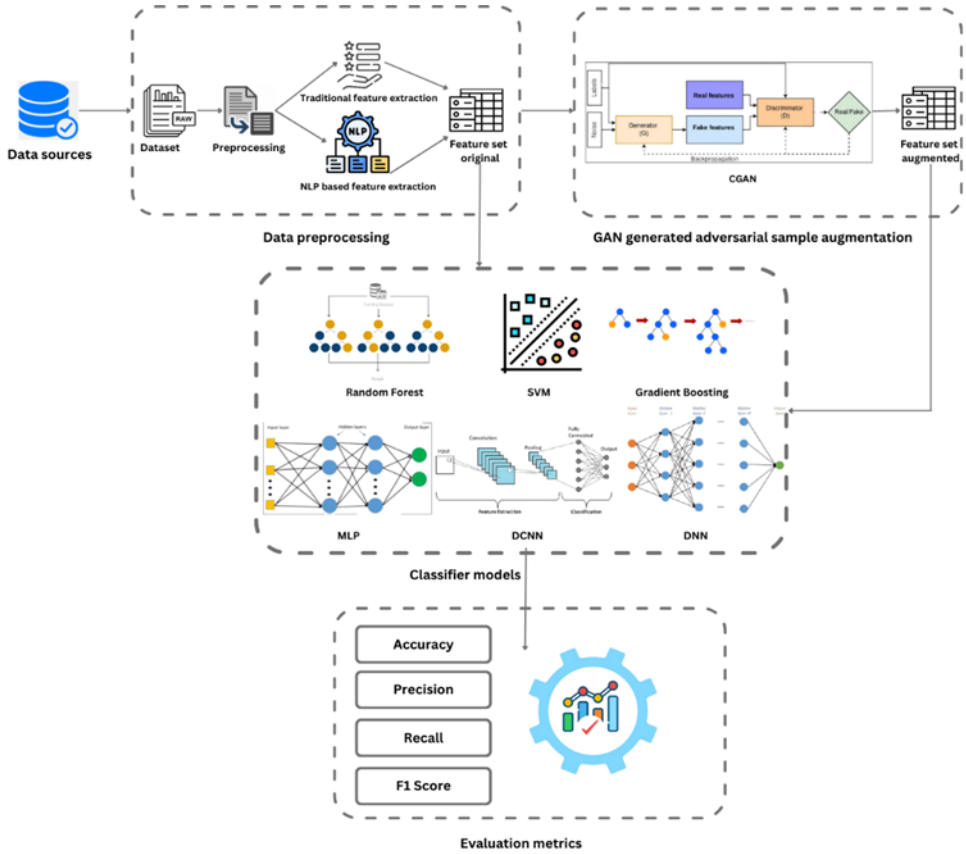

Figure 2: Architecture of malicious classifier model

Multiple supervised ML and DL models as described below are employed for a comparative analysis on both the original and augmented datasets.

- Random Forest is a classification method that employs an ensemble approach, where multiple decision tree classifiers are trained simultaneously on various subsets of the data. The final prediction is made by aggregating the results through majority voting or averaging [19].
- Gradient Boosting just like Random Forest is an ensemble classification model that uses a sequence of individual models typically decision trees where each model corrects the errors of the preceding model to give the final classification [19].
- A Support Vector Machine (SVM) is a supervised learning algorithm used to separate two classes by identifying the optimal hyperplane that maximizes the margin between the nearest data points of each class [22]. Support Vector Classifier (SVC), a type of SVM is used with the Radial Basis Function (RBF) kernel for handling non-linear decision boundaries.
- Multilayer perceptron (MLP) is a feed-forward Artificial Neural Network (ANN) that captures complicated non-linear relationships in data through hidden layers [19]. The MLPClassifier implemented is a fully

connected network consisting of an input layer, three hidden layers using ReLU activation, and an output layer. The model uses L2 regularization and Adam optimizer that increases accuracy for augmented dataset.

- Deep neural network (DNN): DNN is an extension of the MLP with more hidden layers, enabling the model to detect more diverse and complicated patterns in the data. The DNN implemented consists of an input layer, followed by four deep blocks, each containing a dense layer with LeakyReLU activation, BatchNormalization, L2 regularization (weight decay) and Dropout to enhance model robustness, especially against adversarial samples. Softmax activation is used in the output layer. Learning Rate Scheduler (ReduceLROnPlateau) and the Early Stopping callback is added to ensure better convergence. Adam optimizer with learning rate set as 0.001 and sparse categorical cross-entropy loss function is used.
- Dynamic convolution neural network (DCNN): DCNN is a type of CNN that consists of convolutional layers, pooling layers and fully connected layers suitable for sequential data analysis [19]. DCNN is implemented using three convolutional layers using ReLU activation with different kernel sizes followed by Dropout, MaxPooling and BatchNormalization layers making the model architecture dynamic and flexible to diverse input data. Softmax activation is used in the output layer. The model is compiled similarly to the DNN, using sparse categorical cross-entropy loss and Adam optimizer.

IV. DATASET USED

The blacklist used for detection of malicious IP addresses is sourced from a public GitHub repository named IPSum by Miroslav Stampar [21]. IPsum is a threat intelligence resource derived from over 30 distinguished publicly accessible lists of suspicious and malicious IP addresses. The data is automatically collected, processed daily, updated in this repository and displayed in the decreasing order of the number of occurrences provided by the dataset. Malicious URLs dataset [11] from Kaggle is used. It is a collection of a huge dataset of 651191 URLs, which consists of 428103 benign URLs, 96457 defacement URLs, 94111 phishing URLs, and 32520 malware URLs extracted from various sources as shown in figure 3 (a). The sources include Faizan git repo for benign data, URL dataset (ISCX-URL-2016), Phishtank dataset and PhishStorm dataset for malware and phishing data. The URLs of the defacement class are removed since it cannot be detected solely from HTTP headers. After removal of duplicate values, the dataset obtained consists of 545811 URLs with composition as Benign: 78.47%, Phishing: 17.25% and Malware: 4.33%. This dataset is highly unbalanced, so additional phishing URLs are added from Phishing URL dataset by Mendeley [7] as shown in figure 3(b).

```
type
benign        33231
defacement     8269
phishing       2698
malware        1089
Name: count, dtype: int64
```

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement
5	http://buzzfil.net/m/show-art/ils-etaient-loin...	benign
6	espn.go.com/nba/player/_/id/3457/brandon-rush	benign
7	yourbittorrent.com/?q=anthony-hamilton-soulife	benign
8	http://www.pashminaonline.com/pure-pashminas	defacement
9	allmusic.com/album/crazy-from-the-heat-r16990	benign
10	corporationwiki.com/Ohio/Columbus/frank-s-bens...	benign
11	http://www.ikenmijnkunst.nl/index.php/expositi...	defacement
12	myspace.com/video/vid/30602581	benign
13	http://www.lebensmittel-ueberwachung.de/index....	defacement

Figure 3: (a) Snapshot of Kaggle dataset [11] URL dataset

```
type
phishing    54807
Name: count, dtype: int64
```

	url	type
0	https://docs.google.com/presentation/d/e/2PACX...	phishing
1	https://btttelecommunniccatiion.weeblysite.com/	phishing
2	https://kq0hgp.webwave.dev/	phishing
3	https://brittishtele1bt-69836.getresponsesite....	phishing
4	https://bt-internet-105056.weeblysite.com/	phishing
5	https://teleej.weebly.com/	phishing
6	https://maryleyshon.wixsite.com/my-site-1	phishing
7	https://chamakhman.wixsite.com/my-site-4	phishing
8	https://posts-ch.buzz/ch/	phishing
9	https://tinyurl.com/bdfpfyur	phishing
10	https://www.msaaezusshubsnsk.top	phishing
11	https://www.msaaezusshubsnk.top	phishing
12	https://www.msaaezuhubsnk.top	phishing
13	https://docs.google.com/presentation/d/e/2PACX...	phishing
14	https://docs.google.com/presentation/d/e/2PACX...	phishing

Figure 3: (b) Snapshot of Phishing by Mendeley [7]

To further balance the dataset, Synthetic Minority Over-sampling Technique (SMOTE) was applied using a hybrid approach for feature extraction. The hybrid approach used for the extraction of features in the proposed malicious classifier model is outlined in figure 4.

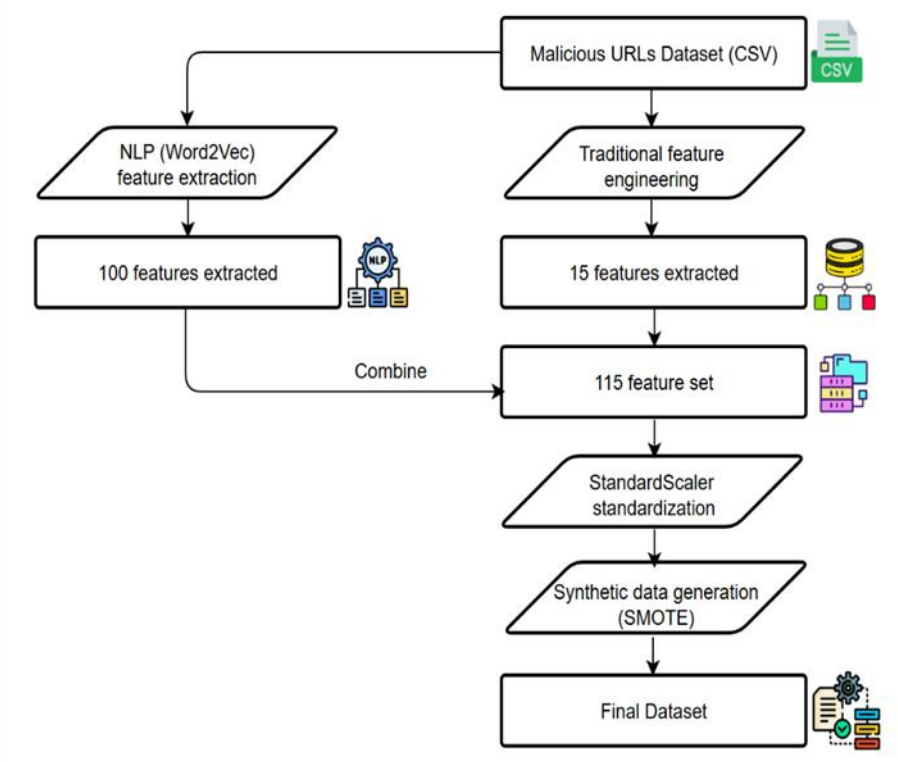

Figure 4: Illustration of data preprocessing

Traditional feature extraction method is used to obtain lexical and structural patterns from the URLs by extracting the following 15 features:

- URL length: The aggregate count of characters in the URL
- Domain length: The aggregate count of characters in the domain part of the URL
- HTTPS vs HTTP: Value is set to 1 for HTTPS protocol and 0 for HTTP
- Dot count: The aggregate count of dots (.) in the URL
- Dash count: The aggregate count of dashes (-) in the URL
- Underscore count: The aggregate count of underscores (_) in the URL
- Question mark count: Total number of question marks (?) in the URL which indicate query parameters
- Special characters count: Total number of special characters in the URL like (! @, #, $, %)
- Digits count: Total number of digits (0-9) in the URL
- IP address presence: Value is set to 1 if the domain is an IP address and 0 if the domain is a hostname
- URL parameters count: Total number of query parameters in the URL
- PHP in URL: Indicates the presence of substring php in the URL with 1 and absence with 0
- HTML in URL: Indicates the presence of substring php in the URL with 1 and absence with 0
- Malicious TLD (Top-Level Domain): Checks if the URL ends with a suspicious TLD for example. xyz, .abc, .ru
- Shortened URL check: Checks if the URL uses any known shortening services for example bit.ly, t.co, ow.ly

Subsequently, NLP based features are generated for capturing the semantic and contextual information from the URLs using tokenization followed by vectorization and aggregation using Word2Vec model. The two feature sets are combined and standardized using StandardScaler as shown in figure 5.

	url_length	domain_length	is_https	dot_count	dash_count	underscore_count	question_mark_count	special_chars_count	digits_count	is_ip_address	...	word2vec_90	word2vec_91	word2vec_92
count	601041.000000	601041.000000	601041.000000	601041.000000	601041.000000	601041.000000	601041.000000	601041.0	601041.000000	601041.000000	...	601041.000000	601041.000000	601041.000000
mean	56.188420	4.595841	0.108181	0.418038	1.472780	0.380367	0.158134	0.0	5.666732	0.020872	...	-0.047791	-0.148368	0.626843
std	66.238354	10.674958	0.310609	0.912738	2.943777	1.215876	0.398059	0.0	13.910584	0.142956	...	0.125346	0.199962	0.356876
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	...	-2.353894	-1.075473	-1.741812
25%	31.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	...	-0.128359	-0.278197	0.471820
50%	43.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	1.000000	0.000000	...	-0.056238	-0.195867	0.698081
75%	68.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	7.000000	0.000000	...	0.025100	-0.069372	0.855715
max	25523.000000	236.000000	1.000000	26.000000	88.000000	79.000000	20.000000	0.0	3413.000000	1.000000	...	0.717686	1.485479	2.026401

8 rows × 115 columns

Figure 5: Snapshot of final generated dataset

After application of SMOTE for synthetic data generation, a balanced dataset is achieved for implementation. To make the proposed model robust against adversarial attacks, adversarial samples are generated using Generative Adversarial Network (GAN) and augmented to the dataset. The model's performance is compared on both the original and augmented dataset. Generator and discriminator are the two primary components of GAN. Conditional GAN (CGAN), one of the types of GAN is used in the proposed model since class labels namely phishing, malware, benign are one-hot encoded and added to the generator and discriminator along with the random noise as conditional parameters. This makes sure that the synthetic adversarial samples generated are corresponding to the features of the given classes. The generator which is responsible for the synthetic data generation and the discriminator which is responsible for classification of the input URL features as real or fake are both implemented using two layers with Rectified Linear Unit (ReLU) activation that are fully connected, followed by an output layer. The generator uses batch normalization with Tanh activation in the output layer while the discriminator uses activation and dropout with sigmoid activation in the output layer. The Adam optimizer, set with a learning rate of 0.0002 along with binary cross-entropy loss is leveraged for training both the generator and the discriminator which is then combined to give the required GAN model.

V. EXPERIMENTAL RESULTS

Total 115 features generated dataset is used and various models are applied for comparison as described in Section III and IV. The adversarial samples were generated using different epoch values for GAN and Random Forest classifier was used to determine which augmented dataset gave the best detection performance. The comparative analysis for GAN epoch 50,70,100 and 120 is as depicted in figure 6.

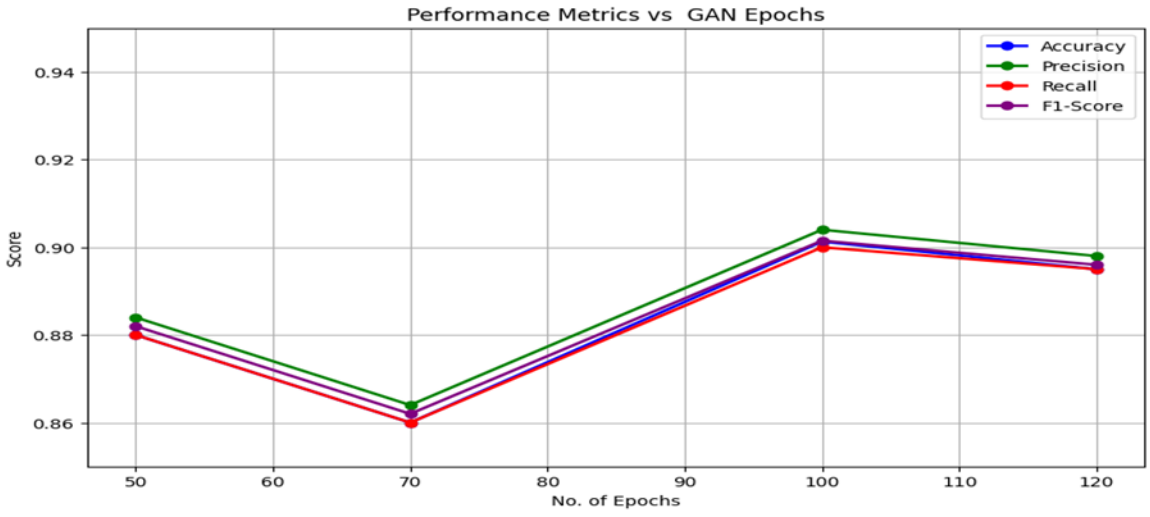

Figure 6: Comparative analysis of GAN for various epochs

The augmented dataset generated using GAN with 100 epochs is considered for further testing of models. The performance of the supervised ML models – Random Forest, Gradient Boosting, SVM and DL model-MLP was verified using cross-validation with 5 folds while sparse categorical cross-entropy loss function was used for DNN and DCNN. K fold cross validation is not used for DNN and DCNN as training these models is computationally expensive, and repeating this process for each fold significantly increases the training time. Comparison of

performance metrics for the original and augmented datasets across different numbers of estimators (70, 100, and 150) for Random Forest classifier and Gradient Boosting classifier is given in table 2 and table 3. The results of SVM classifier are given in table 4.

Table 2: Result of Random Forest Classifier

Random Forest Classifier								
Dataset	**No. of Estimators**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**CV Mean**	**CV Std**	**Training Time (s)**
Original	**70**	0.9037	0.9063	0.9036	0.9041	0.8944	0.0025	11.476
	100	0.9027	0.9055	0.9026	0.9030	0.8945	0.0030	15.864
	150	0.9067	0.9091	0.9065	0.9070	0.8954	0.0032	24.413
Augmented	**70**	0.8997	0.9022	0.8995	0.8999	0.8925	0.0071	12.267
	100	0.9023	0.9052	0.9022	0.9025	0.8950	0.0068	16.050
	150	0.9020	0.9047	0.9020	0.9022	0.8940	0.0082	23.715

Table 3: Result of Gradient Boosting Classifier

Gradient Boosting Classifier								
Dataset	**No. of Estimators**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**CV Mean**	**CV Std**	**Training Time (s)**
Original	**70**	0.8737	0.8745	0.8735	0.8735	0.8753	0.0029	155.839
	100	0.8810	0.8816	0.8809	0.8808	0.8826	0.0040	219.257
	150	0.8930	0.8938	0.8929	0.8929	0.8891	0.0035	324.887
Augmented	**70**	0.8740	0.8750	0.8739	0.8737	0.8728	0.0104	151.401
	100	0.8840	0.8849	0.8838	0.8838	0.8794	0.0106	218.224
	150	0.8923	0.8935	0.8923	0.8923	0.8853	0.0106	328.505

Table 4: Result of SVM Classifier

SVM Classifier							
Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**CV Mean**	**CV Std**	**Training Time (s)**
Original	0.8880	0.8904	0.8879	0.8884	0.8788	0.0079	43.271
Augmented	0.8870	0.8895	0.8869	0.8874	0.8756	0.0109	44.225

According to the results obtained, Random Forest gives the best performance followed by SVM. However, Random Forest and SVM are not adaptable to noisy or augmented data reducing the accuracy of these models on augmented data. Gradient Boosting due to its sequential learning process adapts better to the adversarial samples in the augmented data. The experimental results of MLP for different max iterations is given in table 5. DNN and DCNN is also applied with various epochs to determine which DL model is best suitable for classification as shown in table 6 and table 7.

Table 5: Result of MLP Classifier

MLP Classifier								
Dataset	**No. of max iterations**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**CV Mean**	**CV Std**	**Training Time (s)**
Original	**100**	0.9200	0.9208	0.9187	0.9197	0.9106	0.0032	127.996
	150	0.9206	0.9214	0.9192	0.9203	0.9121	0.0019	132.925
	200	0.9203	0.9209	0.9189	0.9199	0.9051	0.0035	133.221
	300	0.9205	0.9211	0.9191	0.9201	0.9042	0.0035	143.898
Augmented	**100**	0.9280	0.9287	0.9263	0.9275	0.9067	0.0072	129.432
	150	0.9319	0.9325	0.9298	0.9311	0.9118	0.0074	135.672
	200	0.9260	0.9268	0.9242	0.9258	0.9033	0.0066	138.992
	300	0.9263	0.9270	0.9245	0.9257	0.9047	0.0064	147.223

Table 6: Result of DNN

DNN							
Dataset	**No. of Epochs**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Test Loss**	**Training Time (s)**
Original	**70**	0.9283	0.9298	0.9275	0.9286	0.2956	150.003
	100	0.9286	0.9302	0.9279	0.9290	0.2865	188.059
	150	0.9297	0.9315	0.9288	0.9301	0.2933	289.547
	200	0.9308	0.9327	0.9296	0.9311	0.2746	380.003
	500	0.9327	0.9342	0.9319	0.9330	0.2789	874.396
Augmented	**70**	0.9375	0.9412	0.9361	0.9386	0.3669	177.265
	100	0.9410	0.9453	0.9389	0.9421	0.3797	216.172
	150	0.9440	0.9482	0.9421	0.9453	0.3694	338.159
	200	0.9433	0.9471	0.9413	0.9441	0.3425	412.107
	500	0.9425	0.9458	0.9403	0.9433	0.3502	992.326

Table 7: Result of DCNN

DCNN							
Dataset	**No. of Epochs**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Test Loss**	**Training Time (s)**
Original	**70**	0.9083	0.9102	0.9075	0.9088	0.2831	1552.439
	100	0.9105	0.9119	0.9093	0.9106	0.2796	1764.167
	150	0.9113	0.9128	0.9106	0.9117	0.2757	1825.569

Augmented	**70**	0.8927	0.8945	0.8920	0.8932	0.2956	1649.627
	100	0.8953	0.8978	0.8941	0.8960	0.2927	1758.058
	150	0.8932	0.8949	0.8924	0.8936	0.2890	1822.648

It is observed that DNN achieved the highest results on both the original and GAN- augmented datasets. MLP showed reasonable performance while DCNN consistently underperformed, suggesting it is less suited for the task. The augmented datasets led to a drop in DCNN accuracy due to the increased diversity and complexity of the samples. In contrast, MLP and DNN showed improved accuracy on the augmented dataset with appropriate architecture and hyperparameter tuning which indicates the improvement of the models' robustness to adversarial URL samples making it more equipped for real-world applications. Models with their corresponding parameters that achieved the best performance are listed below along with the visual comparison of the accuracy achieved by them in figure 7.

- Random Forest with 150 estimators
- Gradient Boosting with 100 estimators
- SVM with rbf kernel
- MLP with 150 maximum iterations
- DNN with 150 epochs
- DCNN with 100 epochs

```
Initializing Word2Vec model...
Word2Vec model initialized successfully.

URL: https://paypa1.com/authenticate
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 121ms/step

 Classification Results:
Original Model Prediction: Benign
Augmented Model Prediction: Phishing
--------------------------------------------------

URL: https://sh0p.legitimate-store.com/products-special_offer.html
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step

 Classification Results:
Original Model Prediction: Benign
Augmented Model Prediction: Phishing
--------------------------------------------------

URL: https://secure-downloads.com/software/download.exe
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step

 Classification Results:
Original Model Prediction: Benign
Augmented Model Prediction: Malware
--------------------------------------------------

URL: https://micr0soft.com/security-update
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step

 Classification Results:
Original Model Prediction: Benign
Augmented Model Prediction: Phishing
--------------------------------------------------

URL: https://e8ay.com/dashboard
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step

 Classification Results:
Original Model Prediction: Benign
Augmented Model Prediction: Phishing
--------------------------------------------------
```

Figure 7: Accuracy comparison of implemented models on original and augmented dataset

The experimental results show that the DNN with 150 epochs achieved the highest accuracy 92.97% on the original dataset and 94.40% on the augmented dataset highlighting its capacity to capture complex patterns as presented in figure 8.

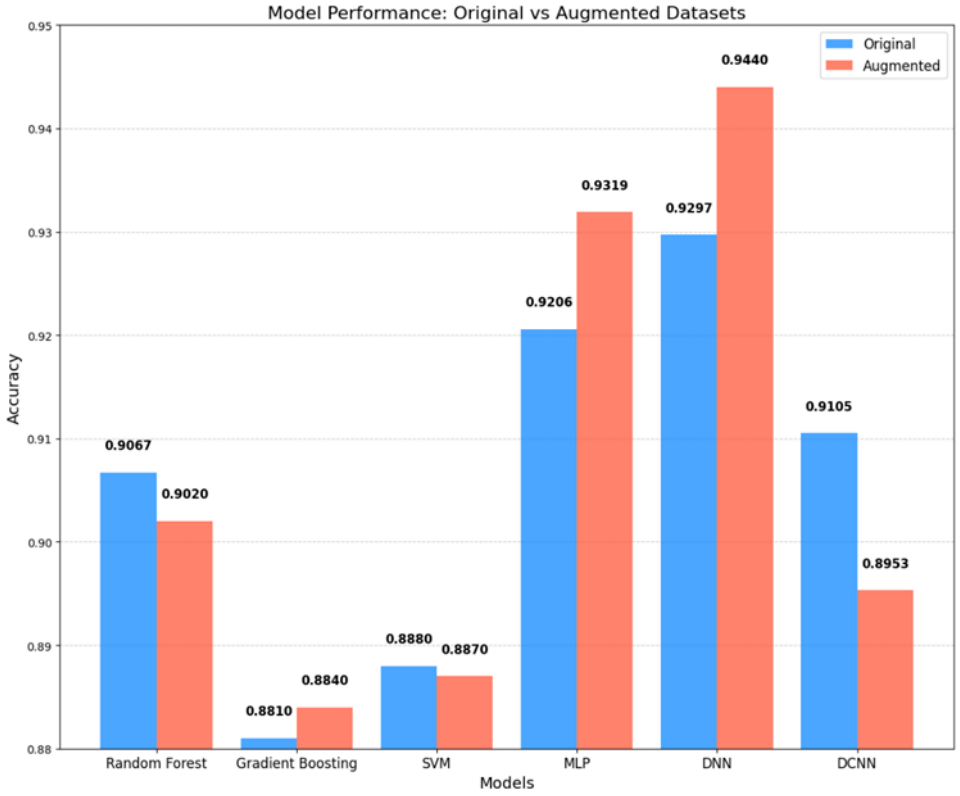

Figure 8: Classification result comparison of DNN trained on original and augmented dataset

Once the model classifies a URL as phishing or malware, it flags the HTTP header containing the URL and based on that the HTTP header attack is detected. For example, if the URL in the Host header is flagged as phishing, the system detects it as a "Phishing attack via Host header injection". Similarly, if the URL is in the Referer header and flagged as malware, it's recognized as a "Malware attack via Referer forgery."

VI. CONCLUSION

This study intercepts HTTP request headers and enables the user to modify these headers. A dual approach for detecting HTTP header injection attacks via the modified headers is proposed. Initially, signature-based attacks are identified, followed by the detection of advanced attacks using malicious URLs through a comprehensive machine learning model. The study leverages a large and balanced dataset retrieved from various sources, with features extracted using a combination of traditional feature engineering and NLP-based methods through Word2Vec. Robustness is further enhanced by augmenting adversarial samples using GAN, and a comparative analysis of classification models-Random Forest, Gradient Boosting, SVM, MLP, DNN, and DCNN was conducted. When epoch is 150, DNN and MLP showed increase in the accuracy for augmented dataset. This proposed framework has significant real-world applications, as it can be seamlessly integrated into web application to strengthen its security by detecting and mitigating phishing and malware attacks through HTTP header analysis. Additionally, its capability to intercept and modify HTTP headers via mitmproxy provides a valuable learning tool for studying various methods of HTTP header injection attacks.

This research is primarily focused on analyzing and detecting attacks through HTTP request headers. Future work could expand the scope to include HTTP response headers, such as Location redirection attacks, which are commonly exploited in phishing and malware campaigns. Furthermore, the framework can be extended to study attacks requiring deeper analysis of complete HTTP traffic patterns. Reinforcement Learning approaches could also be explored to dynamically adapt to new attack vectors while optimizing computational performance and improving model accuracy.

REFERENCES

[1] Ali N. (2023, November 8). What is HTTP response header injection. Beagle Security Blog.https://beaglesecurity.com/blog/vulnerability/http-response-header-injection-found.html

[2] auth0.com. (n.d.). JWT.IO - JSON Web Tokens Introduction. JSON Web Tokens - jwt.io. https://jwt.io/introduction

[3] Buber, Ebubekir & Diri, Banu & Sahingoz, Ozgur. (2018). NLP Based Phishing Attack Detection from URLs. 10.1007/978-3-319-76348-4_59.

[4] Chiarelli, A. (2020, December 8). Preventing HTTPS downgrade attacks. Auth0 Blog. https://auth0.com/blog/preventing-https-downgrade-attacks/

[5] J. Gao, J. Lanchantin, M. L. Soffa and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in Proc.—2018 IEEE Symp. on Security and Privacy Workshops, San Francisco, CA, USA, pp. 50–56, 2018.

[6] M. Grill and M. Rehak, "Malware detection using HTTP user-agent discrepancy identification," 2014 IEEE International Workshop on Information Forensics and Security (WIFS), Atlanta, GA, USA, 2014, pp. 221-226, doi: 10.1109/WIFS.2014.7084331.

[7] KAITHOLIKKAL, JISHNU K S; B, Arthi (2024), "Phishing URL dataset", Mendeley Data, V1, doi: 10.17632/vfszbj9b36.1

[8] Karim, Abdul & Shahroz, Mobeen & Mustofa, Khabib & Brahim Belhaouari, Samir & Joga, S Ramana Kumar. (2023). Phishing Detection System Through Hybrid Machine Learning Based on URL. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3252366.

[9] Laughter, Ashley & Omari, Safwan & Szczurek, Piotr & Perry, Jason. (2021). Detection of Malicious HTTP Requests Using Header and URL Features. 10.1007/978-3-030-63089-8_29.

[10] Ma, Justin & Saul, Lawrence & Savage, Stefan & Voelker, Geoffrey. (2009). Identifying suspicious URLs: An application of large-scale online learning. Proceedings of the 26th International Conference on Machine Learning, ICML 2009. 86. 10.1145/1553374.1553462.

[11] Malicious URLs dataset. (2021, July 23). Kaggle. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

[12] J. McGahagan, D. Bhansali, M. Gratian and M. Cukier, "A Comprehensive Evaluation of HTTP Header Features for Detecting Malicious Websites," 2019 15th European Dependable Computing Conference (EDCC), Naples, Italy, 2019, pp. 75-82, doi: 10.1109/EDCC.2019.00025.

[13] MIZUNO, Sho & HATADA, Mitsuhiro & Mori, Tatsuya & Goto, Shigeki. (2018). Detecting Malware-Infected Devices Using the HTTP Header Patterns. IEICE Transactions on Information and Systems. E101.D. 1370-1379. 10.1587/transinf.2017EDP7294.

[14] Mozilla Contributors. (n.d.). HTTP headers - authentication. MDN Web Docs. Mozilla. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers#authentication

[15] Nair, S. V. (2024, May 14). Reflected Cross Site Scripting. Beagle Security. Retrieved from https://beaglesecurity.com/blog/vulnerability/reflected-xss.html

[16] Nidecki, T. A. (2021, September 13). What is HTTP header injection.The Acunetix Blog. https://www.acunetix.com/blog/web-security-zone/http-header-injection/

[17] Rasheed, Bader & Kazmi, S.M. & Hussain, Rasheed & Jalil Piran, Md & Suh, Doug. (2021). Adversarial Attacks on Featureless Deep Learning Malicious URLs Detection. Computers, Materials & Continua. 68. 921-939. 10.32604/cmc.2021.015452.

[18] Reyes-Dorta, N., Caballero-Gil, P. & Rosa-Remedios, C. Detection of malicious URLs using machine learning. Wireless Netw 30, 7543–7560 (2024). https://doi.org/10.1007/s11276-024-03700-w

[19] Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI. 2, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

[20] Shukla, Sanjeev & Misra, Manoj & Varshney, Gaurav. (2023). HTTP header based phishing attack detection using machine learning. Transactions on Emerging Telecommunications Technologies. 35. 10.1002/ett.4872.

[21] Stamparm. (n.d.). GitHub - stamparm/ipsum: Daily feed of bad IPs (with blacklist hit scores). GitHub. https://github.com/stamparm/ipsum.git

[22] Support Vector machine. (2023, December 27). IBM. Retrieved December 12, 2024, from https://www.ibm.com/topics/support-vector-machine

[23] Suryawanshi, Tushar. "Understanding Host Header Injection Attacks and How to Prevent Them." Medium, 13 Apr. 2023, https://medium.com/@tushar_rs_/understanding-host-header-injection-attacks-and-how-to-prevent-them-60588cd34b8b.

[24] W. Tao, Y. Shunzheng, and X. Bailin, "A novel framework for learning to detect malicious web pages," In Proc. 2010 International Forum on Information Technology and Applications, vol. 2. 2010, pp. 353-357.

[25] Tiefeng, Wu & Wang, Miao & Xi, Yunfang & Zhao, Zhichao. (2022). Malicious URL Detection Model Based on Bidirectional Gated Recurrent Unit and Attention Mechanism. Applied Sciences. 12. 12367. 10.3390/app122312367.

[26] Vanhoenshoven, Frank & Nápoles, Gonzalo & Falcon, Rafael & Vanhoof, Koen & Köppen, Mario. (2016). Detecting Malicious URLs Using Machine Learning Techniques.

[27] Wang, Zhiqiang & Ren, Xiaorui & Li, Shuhao & Wang, Bingyan & Zhang, Jianyi & Yang, Tao. (2021). A Malicious URL Detection Model Based on Convolutional Neural Network. Security and Communication Networks. 2021. 1-12. 10.1155/2021/5518528.

[28] L. Xu, Z. Zhan, S. Xu, and K. Ye, "Cross-layer detection of malicious websites," In Proc. Third ACM conference on Data and Application Security and Privacy, 2013, pp. 141–152.