Problem 7: Predictive modelling for phishing detection

Data Science Capstone

Project made by Lena Handschuh and Rădulescu Carla Gabriela

Submission date: 16/06/2025

Table of contents

Contributors:

- Lena Handschuh 50%
- Carla-Gabriela Rădulescu 50%

Abstract

One of the most used forms of cyberattacks is represented by phishing. Technology is increasingly used by a lot of people over the world, and unfortunately most of them are not aware of possible risks. They are for example tricked into revealing sensitive information through fake websites. As we studied during this research, many detection models rely on outdated or limited datasets, which limits their ability to identify new threats effectively, putting many users in danger.

The purpose of this project is to use both historical and contemporary datasets and to analyze their performance. Following the Machine Learning (ML) pipeline, we first conducted preprocessing steps to clean and scale our data to achieve good results. After we explored it, we trained four machine learning models for each dataset separately, and for a merged version of both, using the following classifiers: Logistic Regression, k-Nearest Neighbors, Decision Tree, and Random Forest. The final evaluation was based on standard metrics such as Accuracy, Precision, Recall, and F1-Score. Across all experiments, Random Forest consistently achieved the highest performance.

Our final findings highlight the importance of using diverse features, which are mostly found in newer datasets. They can improve modern phishing detection systems, protecting people from dangerous threat actors.

Introduction

This project will focus on the development process of enhanced phishing detection models. The datasets used are both contemporary and historical, which makes for a better and improved classification accuracy throughout our work across different phishing techniques as well.

Our first objective is to integrate our datasets, preprocess and standardize features. We will then train our predictive models, which will be able to differentiate between phishing and legitimate samples. We will then compare them to see how well they performed. The final plan is to analyze both datasets separately and then analyze the merged version as well.

Problem statement

One of the most dangerous ways to attack people in an online environment is phishing, a well-known form of cybercrime. Over the years, technology has increasingly evolved, coming along with benefits and dangers. Those with cyber-knowledge, willing to "hunt" people for their personal gain, trick individuals and organizations into revealing sensitive information. As phishing tactics are becoming more and more creative, many existing detection models become outdated or ineffective.

A big issue is strongly related to the reliance on old or limited data sets, that nowadays do not reflect current attack patterns. Our project is meant to address this problem by integrating contemporary and historical datasets to build models that can generalize better. Another purpose is to make it capable of detecting a wider range of phishing techniques, that can help protect people and their personal data.

Objectives

Main objectives and tasks they involve:

- Define the purpose and challenges of phishing detection across evolving techniques
    - Identify gaps in existing phishing detection models
    - Formulate research questions
    - Explain the importance of using two datasets from different time periods
- Data ingestion
    - Download both datasets
    - Inspect them
    - Load data into a working environment
- Clean and standardize datasets
    - Standardize names and formats
    - Encode categorical variables
    - Normalize numeric features
    - Handle inconsistencies
- Split data
    - Ensure class balance
- Train models
    - Choose the machine learning algorithm
    - Train models for each dataset
    - Train a model using both datasets
- Evaluate model's performance and generalization
    - Compare models based on: Accuracy, Precision, Recall and F1 Score
    - Generate confusion matrices

    - Analyse results
- Performance monitoring
    - Document the results
    - Make a conclusion, discussing the limitations and possible future improvements
- Final submission and self-reflection

Data

The older data set focuses on a heuristic-based approach to finding features that are common in phishing websites. It groups features into four categories: Address Bar based Features, Abnormal based Features, HTML and JavaScript based Features and Domain based Features.

Address bar-based features include tactics along the lines of changing up the address bar to conceal any suspicious naming conventions or suspicious domain names in general, e.g. making the URL longer to hide dubious parts of a domain or adding suffix or prefix divided by a dash symbol to make it seem like a legitimate website.

Abnormal based Features include abnormal activities that usually wouldn't occur on a legitimate website such as loading external objects from another domain instead of the same domain or suspicious hostnames in URL doesn't match up with its identity.

HTML and JavaScript based features influence the user experience directly and include activities such as implementing pop-up windows or disabling hovering over links to preview them.

Domain based features are linked directly to properties of the website domain such as its age or by the density of its website traffic.

The second data set features a very similar set of phishing identifiers, including URL features, but adds a few updated factors in comparison to the older dataset, such as whether the webpage runs on the more secure HTTPS protocol. The HTML section is a lot more comprehensive in comparison as well and includes more contemporary features, e.g. taking the responsiveness of the webpage into consideration.

Both datasets include similar features while still showing the difference in topicality from the odd feature that has been added to the newer dataset in contrast to the older one.

Background information

Phishing is a type of cybercrime that victimises companies and individuals alike. A phishing attack usually includes a replica of a legitimate website to leverage personal

data, login credentials, bank details and other data. Victims usually get redirected to a phishing website through legit looking emails usually asking them to update their password or information. Sending messages through SMS and using similar means to send messages are also fairly common. Usually, phishing websites can be distinguished from legitimate ones through small typos in domain names or similar anomalies but visually resemble the legitimate website it's trying to replicate.

Recent research focusing on phishing detection, put accent on two important features: lexical and behavioural. Lexical features are based on how the text of the URL looks, while behavioural features or so-called content-based are about what the website does or contains, involving the action of analysing the web page. We noticed that some systems rely on blacklist matching or manual rules written by humans, methods which are slowly becoming outdated. This is where Machine Learning (ML) makes its presence felt, an approach which have shown better adaptability.

Like we mentioned before, one of the main limitations relies on the quality and diversity of the data. Going through the research documents, it was confirmed to us that most of the datasets used in existing work, lack real-world complexity. By using the two datasets you provided, which come from two different time periods, the model we train aims to improve its ability to detect phishing attacks, including newly emerging threats.

Literature Review

To better understand the context of our project, we decided to go through several sources. In this way, we had the chance to observe what Machine Learning approaches were used, and what techniques and challenges emerged.

1. **Phishing URL Detection Using URL Ranking** [1]

The first valuable source we explored was the study done by Feroz and Mengel, in which they proposed a hybrid phishing detection model that combines *clustering*, *classification*, and *ranking*. For the training phase, they used lexical and host-based features such as URL length, IP usage and DNS info, and introduced a Cluster ID as a new feature derived from K-means clustering. The classifier outputs were mapped to Red, Yellow, and Green threat levels, to enhance interpretability. Their use of feature hashing and scalable learning helped achieve a really good accuracy result of 98.46%, highlighting the importance of feature engineering and combining diverse techniques to boost performance. We plan to use a similar idea by comparing models trained with and without extra features.

2. **Detection of Phishing Websites using Machine Learning** [2]

Next, we analysed a study conducted by a team of six researchers. They focused on real-time phishing detection using a Google Chrome extension, combining *blacklist database checking* with *rule-based semantic analysis* of URL content, buttons, and HTML tags. Their rules focused on URL structure, IP usage and mailto attributes, achieving 99.67% accuracy. This is another important study as they proved that basic but explainable features can still yield strong results, outperforming several existing browser-based methods. This perfectly aligns with our decision of using classifiers like Random Forest, by starting from simple URL features and building up complexity using tree-based models.

3. **Towards Improving Phishing Detection System Using Human in the Loop Deep Learning Model** [3]

The third relevant study of our project was conducted by Asiri, Xiao and Alzahrani. They presented PhishTransformer, *a deep learning model* combined with a *human in the loop system*. Over time, their model re-trains itself after collecting data through a browser extension. The results clearly showed that models trained on new data significantly outperformed those trained on outdated data. What they found directly supports our approach of comparing models on datasets from different periods of time.

4. **Improved Phishing Detection using Model-Based Features** [4]

This source focuses not directly on phishing websites but phishing emails and their suspicious features. It utilises the standard machine learning approach of training classifiers and then learning model parameters on training observation data to be able to detect new emails as phishing. This article mentions the importance of scaling and normalising as key steps of the preprocessing process.

5. **PhiDMA – A phishing detection model with multi-filter approach** [5]

This article mentions the use of a multi-filter approach, which means a cascading of filters which enables filtering with the help of filters from different dimensions. Five filters were to be implemented. Similar to our project, these researchers decided to use the scikit-learn library to evaluate performance. The second layer of their model overlaps with some of our URL-based features.

6. **A predictive model for phishing detection** [6]

As the title says, the goal of this project is to train a predictive model to detect phishing websites. This team utilises a Feature Extraction and Selection module (FSM), which selects features on a frequency basis out of research datasets and literature in this field. The general feature categories, out of which the most frequent features were chosen, match perfectly with ours: URL and webpage properties and webpage behaviours.

7. **PhiUSIIL: A diverse security profile empowered phishing URL detection** [7]

This article is directly related to the newer dataset we plan to use for our project. The researchers wanted to focus on an incremental learning approach, which means it keeps on continuously learning without the need for retraining. This means the model can respond to new threats and can "update" itself without having to retrain it. To select features, different algorithms such as XGBoost, Decision Trees and Random Forest are mentioned. This is a very similar approach as ours.

These papers show a wide range of solutions, from clustering and rule-based detection to active learning, that directly support key directions of our project and the goal we aim to achieve.

Methodology

Tools and Software used

We decided to go for Python as our main language, since we hadn't used it before in the context of machine learning. We mainly went for Jupyter Notebooks to give a clear overview of our process and used PyCharm as our main IDE to collaboratively work on it. The main packages of our project are seaborn and matplotlib for plots, pandas for data manipulation and scikit-learn to train and evaluate our models.

Old Dataset

Cleaning and Preprocessing / EDA

We started off by loading the data and doing some simple exploration. We first converted the file from arff to csv, as we struggled a lot to work with its native format. We found out that this dataset includes an ID column which we deemed as unfit for our model and therefore dropped it. We checked for NAs and couldn't find any. We also looked for duplicates, which did come up, but since the dataset contains mostly binary features, a high number of duplicates was expected. Removing them would unnecessarily shrink the dataset, remaining with no useful information. As our target class we chose the 'Result' column as it represents. We also checked for class imbalance and found that there was some imbalance but not to a worrying extent.

New Dataset

Cleaning and Preprocessing

We loaded the data and removed unnecessary columns that are either covered by other features or were non-integer and thus not needed to train our model. We checked for NAs and didn't have any. There are way less duplicates in this set at 800, which is only around 0.4% of the dataset which has more that 200,000 rows. We decided to drop them to improve our model. We then checked our class imbalance, there was some minor

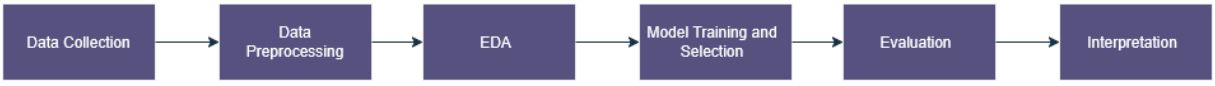
imbalance, but nothing we needed to adjust. The 'label' column represents our target class, 1 represents a legitimate website, 0 a phishing website.

Research questions:

- How effectively can we use machine learning models identify phishing websites based on URL and content features?
- How does integrating historical and contemporary phishing datasets affect the overall predictive performance and generalization of the model?
- What are the differences in phishing characteristics between historical and contemporary datasets?
- Which machine learning algorithms perform best in terms of accuracy, precision, recall, and F1 score?

Analysis Technique

Model Diagram



Steps

Data Collection

We used the two datasets given by the problem statement and downloaded them along their documentation.

Data Preprocessing

We chose to get rid of some irrelevant features and dropped the duplicates for the new dataset.

EDA

We chose some plots according to the dataset to tell us which features are more suspicious or have a high correlation to the result.

Model Training and Selection

We prepared a function to train four different Machine Learning Classifiers. The ones we selected for training were: Logistic Regression, k-Nearest Neighbors also known as k-NN, Decision Tree, and Random Forest. The reason why we chose these, was that we wanted to analyze a mix of linear models, distance-based models, and tree-based models just like in the literature we had conducted. In this way, we could compare performance under different learning approaches.

Therefore, we also applied feature scaling for models that are sensitive to distance such as Logistic Regression and k-NN. Moreover, for the newer dataset we tested a reduced

feature version in two ways: first by removing the highly dominant 'URLSimilarityIndex' column, and later by applying PCA Dimensionality Reduction to keep only 5 principal components.

Evaluation

We evaluated our models using standard performance metrics such as Accuracy, Precision, Recall, and F1 Score. In addition, we also visually analysed the Confusion Matrixes.

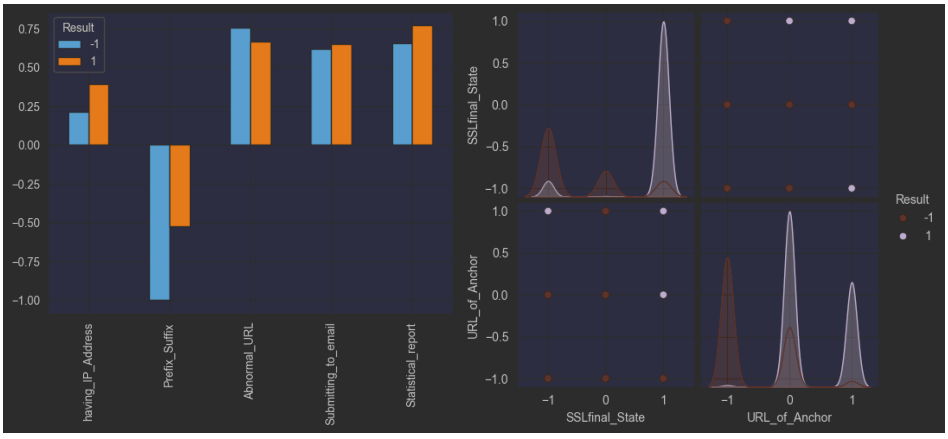
Interpretation

We interpreted the results by comparing strengths and weaknesses of each model. We considered the metrics mentioned above, as well as their ability to generalize across datasets.

Results

Old dataset

We did not spend time cleaning the dataset as it we could not find any null values. However, one initial experiment we performed, was to find the meaning of –1 and 1. We took your advice, and by comparing their average, we found 'Prefix_Suffix' having a perfect value of –1. In our further investigation, looking at the graph below, we noticed that it was the only feature placed in the negative side of the plot, making it suspicious enough. After that, we assumed that the blue bar who reached –1, was more likely to represent a perfect phishing situation. Therefore, we established that suspicious cases are defined by –1, and legitimate ones by 1.



We then visualized feature importance using a Heat Map, followed by feature correlation horizontal bar chart. We checked the possibility of having multicollinearity, which was not the case. Based on the most important features we found, we also build a pair plot which confirmed one more time our assumption of having normal activities at 1, and phishing activities at –1. This plot was attached above, on the right side.

We performed a classic split of 80%-20%. We made sure to include the random seed as we wanted to have the same split every time we ran the code. In addition to this, we took care to correctly stratify the sets, so that bias could not influence the results.

The next part was to train and evaluate the performance. As you can see below, we achieved good results for all the classifiers. Random Forest was the one to perform the best.

	Model	Accuracy	Precision	Recall	F1 Score
0	LogisticRegression	0.928539	0.923441	0.950447	0.936749
1	KNeighborsClassifier	0.949344	0.950121	0.959383	0.954729
2	DecisionTreeClassifier	0.971054	0.970185	0.978067	0.974110
3	RandomForestClassifier	0.974220	0.969600	0.984565	0.977025

New dataset

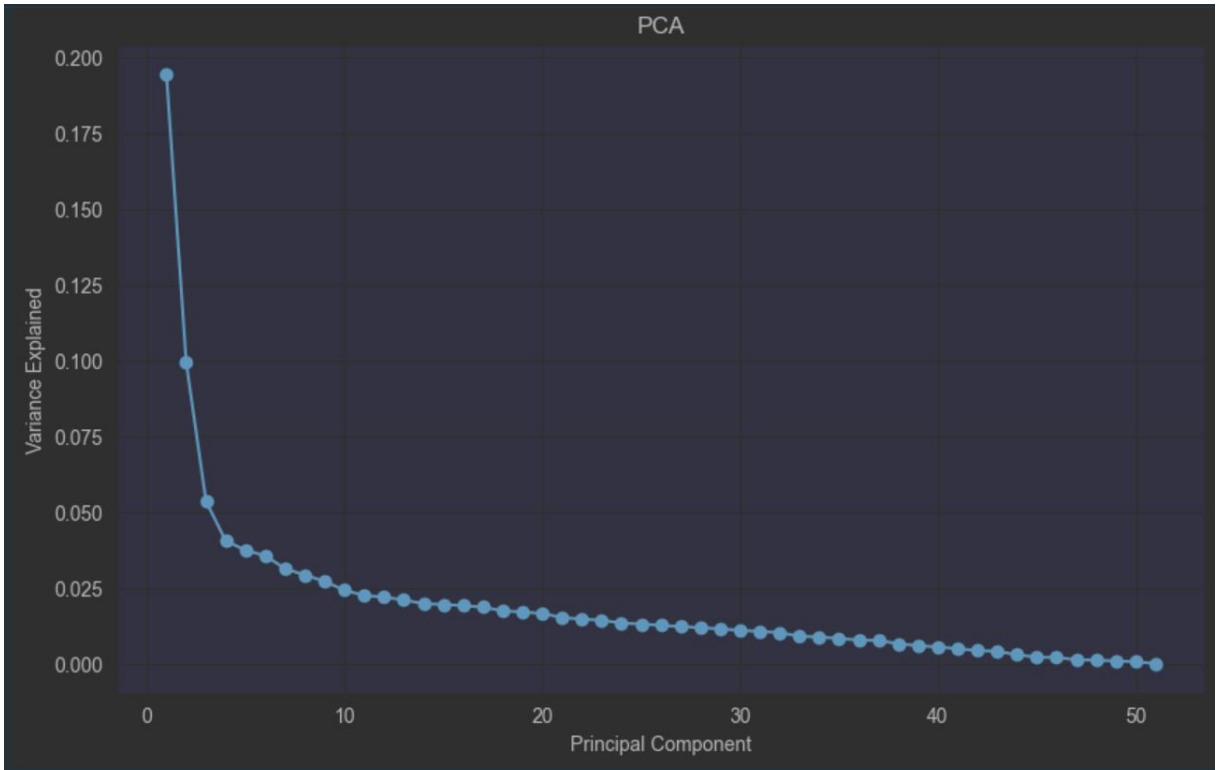
For this dataset, we started by cleaning the data: removing unnecessary columns, dropping duplicates, and verifying class balance which we found to be acceptable.

One important aspect to mention, is that we performed a train-test split of 1%-99%, as we initially got very good results. The size of the dataset was another reason for this decision.

For the training process we wrote a script to train different models on our normal split data and scaled data. Like we already mentioned, we decided to go for Logistic Regression, K-Neighbours, Decision Tree and Random Forest. These were our results:

	Model	Accuracy	Precision	Recall	F1 Score
0	LogisticRegression	0.998324	0.997779	0.999303	0.998540
1	KNeighborsClassifier	0.992925	0.990901	0.996824	0.993854
2	DecisionTreeClassifier	0.991214	0.992935	0.991745	0.992340
3	RandomForestClassifier	0.998728	0.998652	0.999131	0.998892

For this dataset, since the results were so good across the bench, we also decided to conduct PCA and reduced the number of features to the five most important ones. These are the results from that:




```
Top 10 contributing features to PC1:
label                       0.279275
URLSimilarityIndex          0.278715
HasSocialNet                0.238620
HasCopyrightInfo            0.222578
DomainTitleMatchScore       0.221147
HasDescription              0.213577
URLTitleMatchScore          0.211507
SpacialCharRatioInURL       0.201008
```

The elbow seems to be around 3-5 PCs so that's why we went for 5.

Here is the result from the model training with the reduced data for logistic regression and K-Neighbours.

	Model	Accuracy	Precision	Recall	F1 Score
0	LogisticRegression	0.990337	0.989520	0.993685	0.991598
1	KNeighborsClassifier	0.989275	0.986389	0.995041	0.990696
2	DecisionTreeClassifier	0.991214	0.992935	0.991745	0.992340
3	RandomForestClassifier	0.998728	0.998652	0.999131	0.998892

As expected, results slightly decreased but remained very strong. This experiment allowed us to prove that the model has a good performance even with fewer features. Random Forest remained the model which performed the best.

After looking at the initial research paper, which also included model metrics, we concluded that the dataset is extensive enough to substantiate our high results. Even when we applied PCA and reduced the extent to five principal components and even when we changed the training test split, the results were still just as high. Per the results the highest performing model was Random Forest before and after PCA.

Merged datasets

We were between two options: to either perform the cross-dataset evaluation or merge both datasets based on common features. In the end decided to go with the second option as it was considered from the beginning of our research.

We spent quite a lot of time trying to select the correct features. We then tried to rename them and change the target class values of the new dataset, in order to make it align with the older one.

One critical aspect we noticed was that the number of elements from the newer dataset was highly different from the older one. This could introduce bias, and therefore we downsampled it.

We performed a similar exploration as for the previous scripts, visualizing the same plots and using the same function to train our model.

	Model	Accuracy	Precision	Recall	F1 Score
0	LogisticRegression	0.658978	0.708898	0.670810	0.689328
1	KNeighborsClassifier	0.793985	0.818511	0.815557	0.817032
2	DecisionTreeClassifier	0.811850	0.853918	0.803929	0.828170
3	RandomForestClassifier	0.813433	0.854955	0.805934	0.829721

The final performance we achieved was surprisingly good, as you can see in the picture above. Random Forest outperformed again the other classifiers. However, Logistic Regression could not handle the complexity, giving the worst performance of this research.

Discussion

In this section, we would like to briefly answer the research questions of this project.

- **How effectively can we use machine learning models identify phishing websites based on URL and content features?**

Very effectively! All models performed well, especially Random Forest, which consistently achieved the highest performance.

- **How does integrating historical and contemporary phishing datasets affect the overall predictive performance and generalization of the model?**

We can say that merging the datasets improved generalization with only minimal impact on performance. Random Forest adapted well to the combined features and could be further improved through tuning.

- **What are the differences in phishing characteristics between historical and contemporary datasets?**

The main difference we discovered was that the older dataset used more binary features, while the newer one included more varied and continuous data, which boosted performance.

- **Which machine learning algorithms perform best in terms of accuracy, precision, recall, and F1 score?**

Without any doubt, Random Forest! It was consistent, performing well in terms of all the metrics. It was closely followed by Decision Tree. And Logistic Regression had the weakest results overall.

Future work

While our models achieved great performance, there are things to consider such as hyperparameter tuning. For now, we only used default values, but tuning parameters like the number of estimators for Random Forest could help us achieve even better results, especially for the merged datasets.

One more thing we can consider is to test the models we trained on entirely unseen phishing datasets. In this way, we can analyze even better which one generalizes the best, real-word phishing activities.

Conclusion

In conclusion, this project shows the consistency and great work put into the two provided datasets. Both yielded really good results and were well organized, having no NAs and being well balanced.

While the newer dataset had more features and entries, both datasets had comparable metrics. From our model evaluation, the type with the best results for both datasets and our experiment was Random Forest, which consistently outperformed the rest.

We experimented with a merged dataset, which we handled by choosing overlapping features and remapping them. This merge resulted in surprisingly good performance and marks one of the milestones we wanted to reach to offer prediction coverage for older and

newer phishing standards and instances. Another experiment we conducted included experimenting with the new dataset to see if certain measures like PCA and changing splits would affect the great performance of the model. In the end, the performance barely worsened, showing the vast extent of entries making this possible.

In conclusion, this project showed great results all throughout, with especially great performance on random forest classifiers.

References

[1] M. Feroz and S. Mengel, "Phishing URL Detection using URL Ranking," in *2015 IEEE International Congress on Big Data (BigData Congress)*, New York, 2015.

[2] A. Razaque, M. Frej, D. Sabyrov, A. Shaikhyn, F. Amsaad and A. Oun, "Detection of Phishing Websites using Machine Learning," in *2020 IEEE Cloud Summit*, Piscataway, 2020.

[3] S. Asiri, Y. Xiao and S. Alzahrani, "Towards Improving Phishing Detection System Using Human in the Loop Deep Learning Model," in *2024 ACM Southeast Conference*, Marietta, 2024.

[4] A. Bergholz, G. Paaß, F. Reicharzt, S. Strobel and J.-H. Chang, "Improved Phishing Detection using Model-Based Features," in *Conference: CEAS 2008 - The Fifth Conference on Email and Anti-Spam*, Mountain View, 2008.

[5] G. Sonowal and K. K.S., "PhiDMA - A phishing detection model with multi-filter approach," *Journal of King Saud University - Computer and Information Sciences,* pp. 99-112, 2017.

[6] A. Orunsolu, A. Sodiya and A. Akinwale, "A predictive model for phishing detection," *Journal of King Saud University – Computer and Information Sciences,* pp. 232-247, 2019.

[7] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection," *Computers & Security,* 2024.