Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

The applicability of a hybrid framework for automated phishing detection

R.J. van Geest[a], G. Cascavilla[a,*], J. Hulstijn[b], N. Zannone[c]

[a] Eindhoven University of Technology, Jheronimus Academy of Data Science, the Netherlands
[b] University of Luxembourg, Luxembourg
[c] Eindhoven University of Technology, the Netherlands

ARTICLE INFO

Keywords:
Phishing
Phishing detection
Machine learning
Privacy and security
Fraud detection
Security management
Cybersecurity
Cyberthreat Intelligence

ABSTRACT

Phishing attacks are a critical and escalating cybersecurity threat in the modern digital landscape. As cybercriminals continually adapt their techniques, automated phishing detection systems have become essential for safeguarding Internet users. However, many current systems rely on single-analysis models, making them vulnerable to sophisticated bypass attempts by hackers. This research delves into the potential of hybrid approaches, which combine multiple models to enhance both the robustness and effectiveness of phishing detection. It highlights existing hybrid models' limitations that focus primarily on effectiveness while ignoring broader applicability. To address these gaps, we introduce a novel framework explicitly designed for applicability in the real world, which poses the foundation for practical and robust phishing detection architectures. We develop a proof of concept to evaluate its effectiveness, robustness, and detection speed. Additionally, we introduce an innovative methodology for simulating bypass attacks on single-analysis base models. Our experiments demonstrate that the proposed hybrid framework outperforms individual models, displaying higher effectiveness, robustness against bypassing attempts, and real-time detection capabilities. Our proof of concept achieves an accuracy of 97.44% thereby outperforming the current state-of-the-art approach while requiring less computational time. The results provide insights into the multifaceted factors of hybrid models, extending beyond mere effectiveness, and emphasize the importance of holistic applicability in hybrid approaches to address the critical need for robust defenses against phishing attacks.

1. Introduction

Cybercrime is a major problem that keeps growing. Cybersecurity Ventures (Morgan 2020) expects the global cybercrime costs to be over $10 trillion by 2025, which would be the history largest transfer in economic wealth. Among the different types of cybercrimes, phishing is the most common cyber attack according to Abbate (2022). A phishing attack aims to inject malware or obtain sensitive data, such as login credentials, from Internet users. These objectives make phishing a valuable weapon for ransomware attacks and cyber espionage posing a significant threat to all Internet users. On the other hand, many Internet users do not have the experience or skills to distinguish a phishing website from a legitimate one (Peng et al., 2019). This inability keeps increasing since attackers employ increasingly sophisticated techniques to avoid detection (Al Halaseh and Alqatawna 2016; Allodi et al. 2020). Therefore, we need novel solutions to protect Internet users against the dangers of phishing.

Existing research has suggested various solutions for the automated detection of phishing websites, ranging from block-listing (Cao et al. 2008) and heuristic-based methods (Zhang et al. 2007) to more advanced machine learning (Zhang et al. 2007) and deep learning approaches (Do et al. 2022). In particular, the latter have the potential to effectively distinguish phishing websites from legitimate ones. They can classify phishing websites with high accuracy within a short time (Tang and Mahmoud 2021). More interestingly, they can process unstructured data types such as images and texts. This factor is useful for automated phishing detection as the best-performing algorithms analyze only a single website feature, being the URL (Le et al. 2018), HTML (Opara et al. 2020) or a screenshot (Abdelnabi et al. 2020). Although these single analysis-based models achieve high accuracy, they have significant limitations. Research has exposed their vulnerability to bypassing (AlEroud and Karabatis 2020), meaning a phisher can fool the detection algorithm by creating adversarial phishing websites that exploit the algorithm's flaws. The main drawback lies in the fact that these models

* Corresponding author.
E-mail addresses: jobvangeest@gmail.com (R.J. van Geest), g.cascavilla@tue.nl (G. Cascavilla), joris.hulstijn@uni.lu (J. Hulstijn), n.zannone@tue.nl (N. Zannone).

https://doi.org/10.1016/j.cose.2024.103736
Received 6 February 2023; Received in revised form 8 January 2024; Accepted 17 January 2024
Available online 22 January 2024

only analyze a single website feature, making it easier for the phisher to manipulate and bypass the detection.

To address these limitations, Do et al. (2022) propose a hybrid approach combining different models. Each algorithm has strengths and weaknesses, so their combination can leverage the strengths of each model and mitigate their weaknesses. Furthermore, it allows for more comprehensive website analysis, as the different models can each assess a different website feature. Altogether, this would result in more robust phishing detection models.

Some studies have already combined two or multiple single analysis-based models, revealing improved performance (Feng et al. 2020b; Van Dooremaal et al. 2021; Venugopal et al. 2021). Thereby, they show the potential benefit of a hybrid approach. However, these studies measure only the increased accuracy of their approach, i.e., effectiveness. However, they ignore other relevant factors for a phishing detection model to be applicable in the real world.

From the literature, we identify six factors for a real-world model to be applicable in practice (Do et al. 2022; Sahoo et al. 2017): *effectiveness*, *speed of detection*, *scalability*, *adaptation*, *flexibility*, and *robustness*. Together, these factors are the most important for designing and building a real-world automated phishing detection algorithm. Whereas an ideal model would satisfy all these factors, a more realistic one requires a trade-off based on the desired results. These desired results may vary based on the type of application and end-user (Sahoo et al. 2017). From the six applicability factors, the current existing hybrid approaches only measure effectiveness (Feng et al. 2020b; Vecliuc et al. 2021; Venugopal et al. 2021), creating a challenge in gauging the comprehensive applicability of such an approach. Additionally, there is a lack of agreement among these studies regarding the extent to which the hybrid approach improves effectiveness.

This study aims to bridge this gap by assessing the practical applicability of a hybrid approach for detecting phishing pages. We do so by building a framework that focuses on the six factors of applicability. We meet the call for more robust models while also considering the other factors required for a real-world detection algorithm. The goal of this study translated into the following research question:

RQ: To what extent is a hybrid framework suitable for automated phishing detection?

We answer this research question with a two-fold approach. First, we construct a general hybrid framework that takes all factors of applicability into account. Secondly, we build a proof of concept of the hybrid framework and test it on effectiveness, robustness, and detection speed. We test for only these three factors as the other three factors of applicability (scalability, adaptation, flexibility) are, in principle, enhanced by a modular architecture. Furthermore, these factors depend on the factors of the individual models. Therefore, these factors cannot be measured explicitly and fall outside the scope of our experiments. This results in the following sub-questions:

RQ1: How effective is a hybrid framework for automated phishing detection?

RQ2: What is the speed of detection of a hybrid framework to detect phishing in an automated system?

RQ3: To what degree is a hybrid framework for automated phishing detection robust to bypassing efforts?

This study assesses effectiveness as it is the applicability factor used by other studies. Furthermore, effectiveness regards the core task of a phishing detection algorithm: distinguishing phishing websites from legitimate ones. We add robustness, as the literature urges the importance of this applicability factor because of the risk of bypassing (Do et al. 2022). Finally, we measure detection speed. In particular, we are interested in the difference in detection time between the hybrid approach and the individual models incorporated.

Our framework combines predictions of different single analysis-based models. To do so, we use a stacking function. This function takes each model's prediction as input and mathematically combines them into one prediction. By applying various stacking functions to the proof of concept, we determine which stacking function yields the highest applicability of a hybrid framework and works best for detecting phishing websites.

RQ4: Which stacking function performs best in a hybrid framework for automated phishing detection?

This research contributes to the field of automated phishing detection by advancing our understanding of hybrid approaches, assessing their applicability, and evaluating their potential to enhance the robustness of detection models. Fig. 1 shows a schematic overview of the framework. The study introduces a novel bypass simulation method, revealing the hybrid approach's resilience against sophisticated phishing attempts and reducing internet users' vulnerability to such attacks. By examining multiple applicability factors, including robustness and detection speed, the research offers a more comprehensive perspective on the strengths and limitations of hybrid models. We devise and test a proof of concept consisting of three specific deep learning models, each analyzing a different website feature: URL, HTML content, and HTML DOM tree structure. The URL-based model processes the URL as a piece of text. It assesses the interrelationships between the words and the separate characters in the URL (Le et al. 2018). The HTML content-based model uses a similar approach to analyze the HTML code (Opara et al. 2020). For the HTML DOM tree analysis, the model first extracts the DOM tree structure out of the HTML and then analyses this as a text sequence (Feng et al. 2020a). Each model makes a prediction for each website in the test set, which we use as input for various stacking functions. We assess the effectiveness by how well the model classifies websites it has never seen before. The time it takes the proof of concept to make predictions shows us the detection speed. For robustness, we simulate a bypass of one of the incorporated models. We measure the impact this has on the performance of the proof of concept. Based on the results of these tests, we determine the applicability of the proof of concept and the framework. Thus, our framework provides a general basis for applicable and robust phishing detection architectures. Our findings provide practical implications for strengthening cybersecurity defenses and pose the foundation for developing more effective and resilient phishing detection architectures, addressing a critical cybersecurity challenge.

The remainder of the paper is structured as follows. Section 2 introduces the contribution of our research and the related societal impact. Section 3 presents background information on phishing and machine learning and discusses related work on automated phishing detection. In Section 4, we devise the hybrid framework optimized for applicability and explain the implementation of the proof of concept. Section 5 discusses the experiments to evaluate the proof of concept. We discuss the findings of these experiments in Section 6. The paper ends with conclusions and suggestions for future research in Section 7 and Section 8. The dataset used for the experiments and the replication package are available at https://doi.org/10.5281/zenodo.8358925.

2. Research contribution

Our study's primary contribution lies in its multifaceted approach to the evaluation of a hybrid framework for phishing detection, emphasizing real-world applicability and robustness. This research stems from critical gaps identified in existing literature, where prior studies have often fallen short in adequately addressing these dimensions of applicability and robustness in the context of phishing detection.

By conducting a comprehensive evaluation of the hybrid framework, we bridge these gaps and provide concrete evidence that the hybrid ap-

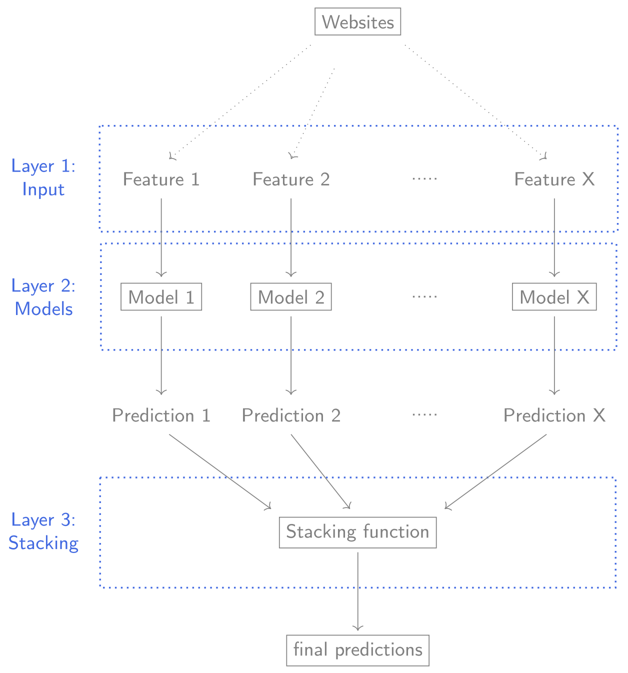

Fig. 1. Schematic overview of the hybrid framework aimed at applicability for automated phishing detection.

proach offers a robust and effective alternative to single-analysis-based models. We go beyond the typical emphasis on effectiveness alone, expanding our assessment to encompass various factors that define real-world utility. In doing so, we contribute to advancing knowledge in the field and offer practical insights for deploying phishing detection systems.

Our research serves as a stepping stone for future investigations that can extend beyond the boundaries of our current study. Specifically, we propose the exploration of additional dimensions of applicability, including flexibility, adaptability, and scalability. These dimensions, often overlooked in previous research, represent crucial factors of real-world applicability. For instance, assessing the framework's flexibility can validate our hypothesis that implementing a stacking function enhances adaptability compared to integrating models into a single neural network. The introduction of online learning, using datasets with evolving behaviors over time, offers a promising avenue to evaluate adaptability, allowing for the creation of more dynamic and resilient systems. Furthermore, investigating scalability, as indicated by Sahoo et al. (2017), is essential to understanding how the framework performs under increased data volumes, a consideration of utmost importance in today's data-rich environment.

In summary, our study highlights the existing gaps in phishing detection research and takes significant strides toward filling them. By focusing on applicability and robustness and conducting a thorough evaluation of a hybrid framework, we contribute to a more comprehensive understanding of hybrid approaches and their real-world potential. Our research serves as a foundation for future studies, offering a roadmap to explore these dimensions further and ultimately enhance the practical deployment of phishing detection systems.

2.1. Societal impact

An automated phishing website detection algorithm can improve the quality of life of an underprivileged or vulnerable part of society by providing a way to avoid being phished. Scammers often create phishing websites to trick people into giving them personal information or money. These websites can be difficult to spot, but an automated phishing website detection algorithm can help identify and protect internet users from being scammed. This can help improve the quality of life of those vulnerable to these scams, as they will be less likely to fall victim to them. With our study, we aim to make these detection models better applicable. Thereby guiding research to better fulfill the societal needs for real-life detection algorithms (Sahoo et al. 2017).

The key contribution to societal impact is the data-backed robustness of this hybrid framework. Where a bypass would disarm a single analysis-based model, the hybrid approach remains to function. This makes internet users less vulnerable to bypassing the efforts of phishers.

3. Background & related work

Phishing is a form of social engineering that uses fraudulent messages to trick people into revealing their personal information or credentials (Abbate 2022). Phishing is a type of online fraud in which scammers send emails or messages that look like they are from a legitimate company or website. These messages often contain a link that takes the victim to a fake website that looks like the real one. The victim is asked to enter personal information on the fake website, such as their credit card number or password. The scammers then use this information to steal the victim's money or identity.

The first relevant method for detecting phishing websites is blocklisting, where a moderator places websites suspected of phishing on a list of websites to be blocked (Cao et al. 2008). However, this only works for already detected websites, while attackers can keep registering new domains and exploiting them until they are compromised. These new and yet undetected phishing websites are called zero days. To detect zero-days, research proposes heuristic-based methods that check multiple aspects of a website, such as hostname and lifetime (Teraguchi and Mitchell 2004). However, these approaches result in many false positives (websites incorrectly classified as phishing). To tackle this, Zhang et al. (2007) developed CANTINA: a heuristic-based method that also analyzes the content of a website. This algorithm recognizes common signs of phishing. For example, linking back to the legitimate website from the phishing website creates a discrepancy between the domains. This approach decreases the false positives. Nevertheless, the heuristics are still relatively straightforward. This makes it easy for an attacker to understand the heuristics and create bypassing methods.

Machine Learning allows for much more sophisticated and powerful classification algorithms. This enables the analysis of a wide range of website features. Xiang et al. (2011) used this method to develop CANTINA+: an algorithm that analyses fifteen different website features obtained from the website's URL and HTML content. Therefore, it was the most comprehensive analysis at that time. Each feature represents a website characteristic that the authors believe indicates phishing. CANTINA+ is one of many machine learning models that use the URL and HTML (Hou et al. 2010; Odeh et al. 2021; Sahingoz et al. 2019) as input. These data types easily translate into functional features such as the length and number of unique symbols. Another commonality among the machine learning approaches is the use of supervised learning. This preference is due to the existence of labeled datasets. Platforms exist that maintain publicly available databases of phishing websites (OpenPhish 2022; PhishTank 2022).

Deep learning, a sub-field of machine learning, takes this concept a step further by incorporating neural networks. One advantage of deep learning is its scalability, which can handle large amounts of data, leading to more accurate predictions (Alom et al. 2019; Ejaz et al. 2023). Another advantage is that deep learning can learn from not linearly separable data. This means the data does not have to be in a specific format. We can input it, for example, as an image or text. Finally, deep learning is less likely to overfit the data. This means that the algorithm will not learn from the noise in the data, which can lead to more accurate predictions (LeCun et al. 2015). Combining these benefits makes deep learning models detect hidden correlations in the data. Moreover, data can be provided to the algorithm with only minor preparation steps.

This is contrary to supervised machine learning methods, which require expert knowledge to select the appropriate features (Do et al. 2022).

URL base models

As a URL is interpretable as a piece of text, we can analyze it with Natural Language Processing. One way to do so is by embedding the URL into a mathematically understandable format. There are two types of URL embedding: character-level and word-level embedding. Character-level embedding considers the URL as a sequence of individual characters, while word-level embedding regards it as a sequence of words. Yang et al. (2021) found that using character-level embedding in combination with Convolutional Neural Networks (CNN) and multiple Random Forest classifiers resulted in a detection accuracy of 99%. However, solely using Character-level CNNs also comes with limitations. This approach is unable to capture semantic or sequential patterns effectively. Hence, structural information contained in words gets lost. As a solution, URLNet integrates the character-level embedding into the word-level embedding (Le et al. 2018). Instead of discarding rare words, it represents them by character-level embedding. Next, it integrates the obtained information back into the Word-level embedding. The benefit is that information of rare words is kept, without raising memory issues. The successful performance and clear reproducibility made URLNet the benchmark model for URL-based phishing detection (Bu and Cho 2021a,b; Bu and Kim 2022; Dutta 2021; Kexin et al. 2021; Maneriker et al. 2021).

HTML based models

Opara et al. (2020) are the first to use the HTML content of a website as input for a Deep Learning algorithm. They feed it as a single text string to their HTMLphish model. Similarly to the previously described URL methods, it consists of a character-level CNN and word-level CNN. The authors obtained a 93% accuracy on their test data. This method is third-party independent and can detect phishing websites regardless of their language. Although HTML content is valuable for phishing detection, it loses the structural information, which is represented in a HTML DOM tree structure. Ouyang and Zhang (2021) use this data type for a Graph Neural Network (GNN). They input the HTML as a graph via its inherent DOM tree structure. This approach results in a phishing detection accuracy of 96%. Feng et al. (2020a) also analyse the HTML DOM tree structure. However, they regard them as a natural language. They use Doc2Vec to learn the structural semantics to detect phishing web pages automatically. Finally, Bilot et al. (2022) analyse the HTML content based on the hyperlink graph structure of a website. Many phishing websites redirect to legitimate websites, so each link pointing to these websites has a different domain. Legitimate websites, on the other hand, typically have many links redirecting to the same domain. This referring difference creates a discrepancy that can be detected.

Limitations

Although Deep Learning shows advantages for the automated detection of phishing websites, it has limitations. First, these models operate in a black-box fashion, making it difficult to understand the reasoning behind the prediction. This problem is compounded when errors occur, as it is hard to diagnose and identify the root cause of an error when the output results are largely uninterpretable (Do et al. 2022).

Furthermore, AlEroud and Karabatis (2020) show that carefully crafted inputs can deceive Deep Learning models. They present a method for bypassing an URL-analysis-based detection model using Generative Adversarial Networks (GANs). The proposed method generates phishing URLs that are visually similar to legitimate URLs, making them difficult to detect. The GAN is trained on a dataset of legitimate and phishing URLs and can generate new phishing URLs that are not in the training set. The generated URLs are then evaluated against a phishing detection system. The results show that the proposed method can bypass the detection of phishing URLs with high success rates. From this, we can conclude that detection algorithms that are not robust against such adversarial attacks will not be applicable in the real world, as hackers will always try to find new methods to bypass protection.

3.1. Applicability

In a hybrid approach, we can combine models in different ways. To find the optimal way, we first determine the goal of the algorithm. In our case, this is applicability, with a focus on robustness. To determine applicability, we look at the current literature. Sahoo et al. (2017) identify five design principles for building a real-life automated phishing detection algorithm:

Accuracy The predictive performance of the algorithm refers to its ability to classify phishing and legitimate websites accurately. Researchers use the term Accuracy to refer to this factor; however, it is important to note that it is a general term for predictive performance and not limited to just the Accuracy metric. Therefore, we refer to this design principle as Effectiveness.

Speed of Detection The time it takes to classify a webpage. An architecture that takes too long is not suitable for live classification within a web browser. Therefore, this requirement greatly influences the usability for end-users. According to Sahoo et al. (2017), real-time detection requires a classification within a couple of milliseconds. We measure the speed of detection by the time it takes the model to make a prediction. This counts from the moment it starts analyzing until it gives an output.

Scalability The architecture's capacity to process large amounts of data. To obtain this, it should prevent memory constraints.

Adaptation The ability to detect and adapt to changes in the data. In particular, the ability to detect new types of phishing pages. Researchers believe better adaptability improves the robustness of the architecture.

Flexibility Flexibility requires the system to allow for easy improvements and extensions. This regards the broadness of the architecture, as well as the performance. New types of data inputs should be implementable easily. Furthermore, the architecture should allow for implementing new developments found in the literature.

While adhering to these principles, we differentiate between *Adaptability*, which focuses on the algorithm's proficiency in identifying and accommodating data changes, and *Robustness*, which pertains to the algorithm's capacity to offset potential flaws or vulnerabilities within its components. We advocate that this distinction provides a more nuanced and precise understanding of a phishing detection algorithm's effectiveness. In particular, we gain the ability to specifically evaluate an algorithm's capability to identify and adjust to changes in the data it encounters. Adaptability, in this context, focuses on the algorithm's agility in recognizing new types of phishing pages and accommodating variations in the data landscape. It essentially measures the algorithm's responsiveness to evolving threats and its capacity to adapt swiftly, reflecting the dynamic nature of the cybersecurity landscape. On the other hand, Robustness assesses the algorithm's resilience against potential weaknesses, vulnerabilities, or adversarial attempts to bypass its defenses. This factor is vital for ensuring the algorithm's ability to maintain its effectiveness even in the face of sophisticated attacks or unforeseen challenges. It measures the algorithm's ability to withstand and counteract flaws in its components, further reinforcing its real-world applicability. In essence, this division provides a more comprehensive view of an algorithm's applicability by considering its adaptability to evolving threats and its robustness in the face of potential vulnerabilities. Therefore, we add Robustness as a sixth, separate factor of applicability.

These principles of applicability serve as valuable guidelines to identify issues and gaps in the current approaches and literature on automated phishing detection. We can pinpoint areas where the literature may fall short by evaluating existing research through the lenses of Effectiveness, Speed of Detection, Scalability, Adaptation, Flexibility and

Table 1
The expected impact of a hybrid approach on current challenges in automated phishing detection. The challenges originate from a study by Sahoo et al. (2017). The impact column shows a hybrid approach's expected impact on these challenges: negative, neutral, or positive. The explanation column motivates the expected impact.

Challenge	Impact	Explanation
High volume and high-velocity data	Neutral	There are no signs that a hybrid approach is more efficient than other approaches.
Difficulty acquiring labels	Neutral	A hybrid approach does not generate new labels.
Difficulty in collecting features	Neutral	This depends on the models implemented in the hybrid approach.
Feature Representation	**Positive**	A hybrid approach can process multiple website features via different implemented models (Feng et al. 2020b).
Concept drifting	Neutral	This depends on the models implemented in the hybrid approach.
Interpretability of Models	**Positive**	Combining different predictions give insights into the influence of the different models (Do et al. 2022).
Adversarial Attacks	**Positive**	Combining multiple models makes it harder for an attacker to deceive the entire system (Venugopal et al. 2021).

Robustness. For instance, if a substantial body of work predominantly emphasizes effectiveness but neglects speed of detection or scalability factors, it suggests a potential research gap in addressing real-time, resource-efficient detection methods. Similarly, if there is limited focus on the risk of bypassing by adversarial attacks, it highlights opportunities for future research to enhance the robustness and practicality of phishing detection algorithms. Therefore, the application of these principles aids in identifying research avenues that can contribute to a more comprehensive and effective automated phishing detection framework.

3.2. Hybrid approach

Do et al. (2022) recommend using a hybrid approach to address multiple challenges in automated phishing detection. This approach combines different models into a single algorithm to optimize their strengths and weaknesses, enabling more robust detection. According to Feng et al. (2020b), it also facilitates comprehensive website analysis by allowing various models to assess different website features websites.

Sahoo et al. (2017) identify various challenges in the automated detection of phishing websites, including the issue of detecting malicious websites despite evasion techniques. Table 1 outlines these challenges and highlights the positive impact of using a hybrid approach. First, this approach leverages the strengths and mitigates the weaknesses of different detection algorithms, enhancing overall capabilities and making it harder for attackers to evade detection. Secondly, it enables comprehensive website analysis by distributing the evaluation of diverse website features among individual models, thereby improving accuracy. Lastly, hybrid approaches enhance robustness against evolving evasion tactics, enhancing cybersecurity capabilities by effectively identifying anomalies and discrepancies employed by attackers and reinforcing the system's resilience.

3.3. Existing hybrid approaches

As discussed in Section 3, existing research has commonly studied the performance of models using just raw URLs (Al-Ahmadi et al., 2022; Chinnasamy et al., 2022; Elsadig et al., 2022; Kim et al., 2022; Ozcan et al., 2023) or HTML content (Aljofey et al., 2022; Ejaz et al., 2023; Lee et al., 2020; Vishva and Aju, 2021). However, based on our studies, experiments, and research, we highlight that the information in different parts of a web page can provide different information and characteristics when detecting phishing pages. The HTML provides the semantics of the page, the URL gives information regarding the web page's internet address, and the DOM (Document Object Model) tree is a tree-like representation of the web page and furnishes details of the backbone of an HTML document, such as tags. Therefore, leveraging all this information is essential to detect phishing.

In assessing the landscape of existing hybrid approaches, we apply the design principles of applicability as established by Sahoo et al. (2017). These principles, presented in 3.1, offer valuable insights into current hybrid models' strengths, weaknesses, and overall limitations, shedding light on their effectiveness in automated phishing detection.

These applicability principles serve as essential guidelines to assist us in pinpointing potential gaps in the current literature or areas where existing research may need to be improved.

Opara et al. (2023) present WebPhish, which implements a deep neural network trained using embedded raw URLs and HTML to detect phishing attacks. WebPhish showed an accuracy of 98.1%. However, WebPhish can only detect zero-day phishing attacks containing known HTML and URL content. If the attack involves manipulation of the web page content, WebPhish cannot recognize the attack as this approach is strictly dependent on the training set. Similarly, Venugopal et al. (2021) conducted a study testing various machine learning models within a hybrid framework. These models analyzed different manually obtained features from the HTML and URL of a website. Notably, each model was evaluated using a different dataset, revealing a substantial variation in accuracy (from 73.8 to 99.98). In their hybrid configuration, they combined the output predictions of these models using a decision tree and assessed them on an ensemble dataset. Surprisingly, this hybrid approach resulted in an accuracy of 95.3%, which fell below the performance of some individual integrated models. This contrasts with our expectation that a hybrid approach improves effectiveness. The only other design principle discussed by the authors is robustness. They claim their framework makes it harder for hackers to infiltrate the system. However, they do not give any proof of that. Another interesting hybrid approach is presented by Aljofey et al. (2022) where the authors leverage URL and HTML features. The proposed approach achieved an accuracy of 96.76%. However, the proposed approach needs a long training session and depends on the English language. Different language causes an error in the classification.

Vecliuc et al. (2021) presented a contrasting outcome on effectiveness. They tested a similar approach on a website's URL, HTML content, and logo. This resulted in a corresponding accuracy of 96.5% for their hybrid model, which is slightly higher than their best-performing individual model, which was the ULR-based model with 96%. Thereby suggesting that a hybrid approach can improve the effectiveness, although it does not seem to make a large difference. Again, no other principles of applicability were discussed.

In contrast to these findings, Feng et al. (2020b) introduced a different perspective with their Web2Vec hybrid model. This model incorporated the URL, HTML content, and DOM tree structure as input features. Their study involved comparing performance across different input feature combinations. While they obtained an accuracy of respectively 94.72%, 97.56%, and 91.71% for the URL, HTML content, and DOM tree structure, they acquired an accuracy of 99.05% for the combination of the three. This achievement was attributed to their method of extracting feature representations and processing them through a CNN and LSTM. Instead of obtaining separate predictions from this and combining these with a stacking function, they combined the neural network layers as input for one larger hidden layer. Hence, this layer contains processed information from each of the input features. This step merges the different models into one large neural network with one output layer that gives the final prediction.

Besides the improvement in effectiveness, it is important to note that the Web2Vec approach also has limitations, primarily its black-box nature, making it harder to understand the reasoning behind a prediction and identify errors when they occur. As bypassing by an attacker is an error of the model, we could deduce from this that this approach is less robust. We cannot verify this as the authors state nothing about the robustness of their model. Another limitation is that its design lacked flexibility. As their hybrid approach consists of one interconnected neural network, it hinders easy adaptation to evolving requirements or the integration of new models or algorithms. Compared to this, the approaches of Venugopal et al. (2021) and Vecliuc et al. (2021) are more flexible, as they use the predictions of individual models, which can be more easily replaced with the predictions of a different model without significant disruption to the overall system. Regarding detection speed, the researchers state that their approach spends the longest running time compared to other approaches. They attribute this to the use of deep learning algorithms. This reasoning indicates that detection speed highly depends on the selection of the individual models. Lastly, Valiyaveedu et al. (2021) revealed that different input features might benefit from distinct models, suggesting a potential avenue for further improvement.

Recent works such as (Lin et al., 2021; Liu et al., 2022) implemented a similar approach. In (Lin et al., 2021), similarly to other less recent works (Abdelnabi et al., 2020; Afroz and Greenstadt, 2011; Wang, 2010), the authors use logos and web pages screenshots from the website to recognize phishing pages. Even if the authors claim that Phishpedia can recognize 87.46% of phishing web pages, it is worth noting the high rate of false positives related to logos looking like a well-known legitimate brand logo. Furthermore, the authors explain how Phishpedia is not suitable for large-scale evaluations due to its high rate of false negatives. The same authors proposed PishIntention Liu et al. (2022), which generates 86.5% less false alerts. The proposed approach receives a URL, screenshot, and HTML code and extracts the Abstract Webpage Layout for detecting brand and credential-taking intentions. However, as specified by the authors, it fails when the web pages implement an uncommon login-keyword. Furthermore, the robustness of this approach drops significantly in front of HTML obfuscation. Lastly, the authors explicitly explain that although this approach has a lower false positive rate than Phishpedia, the problem remains due to logo similarities.

To conclude our literature review, it is important to highlight Random Forest provides the best classifier when adopting a URL or HTML-based approach in the context of phishing page detection (Almousa et al., 2022; Chinnasamy et al., 2022; Liu, 2021; Ripa et al., 2021). However, as mentioned in Chinnasamy et al. (2022); Shah et al. (2022), future research should focus on hybrid approaches and technologies to detect phishing websites more accurately and improve their robustness. Indeed, machine learning-based phishing URL detectors have been extensively proposed and explored. However, the robustness of these models against adversarial manipulation remains unknown. Sabir et al. (2022) unveiled several security vulnerabilities and evaluation challenges of the machine and deep learning models.

Our approach can bypass the limitations of the previous works by considering three different features for the phishing classification: URL, HTML, and DOMtree. These three features improve the overall robustness of our approach, making it able to bypass limitations such as language, training session time, easier to go through the system flow, the system not working as a black box, easier to catch errors and problems, improved the overall robustness against possible bypass. Moreover, our proof of concept provides a wide margin of customization and flexibility. Indeed, although our proof of concept of our framework leverages the three aforementioned features for the detection of phishing websites, it can be easily extended to include other features such as images and meta-data.

3.4. Gaps in literature

While the existing literature provides valuable insights into automated phishing detection and hybrid approaches, we observed several gaps related to various factors of applicability. First, the literature presents conflicting results regarding the effectiveness of hybrid approaches compared to individual models. Venugopal et al. (2021) observed that their hybrid approach underperformed some individual models, while Vecliuc et al. (2021) found a modest improvement in effectiveness. Feng et al. (2020b) demonstrated significant improvement, but their approach lacked transparency. The approaches proposed in (Lin et al., 2021; Liu et al., 2022) suffer from a high rate of false alerts. Ejaz et al. (2023) explicitly mention the need to improve phishing techniques to be robust against advanced evasion techniques. However, their approach based on continual learning suffers from *catastrophic forgetting* when the model is trained on new data. Other works (Alhogail and Alsabih, 2021; Aljofey et al., 2020; Su, 2020; Wei et al., 2020; Xiao et al., 2021) do not consider (nor test) the robustness of their approaches against side attacks during their experiments. Regarding effectiveness, further research is needed to provide a clearer understanding of when and how hybrid models impact this factor.

Furthermore, the speed of phishing detection for a hybrid framework is currently unclear. While Venugopal et al. (2021) mentioned the importance of real-time detection, there is a lack of detailed exploration of the trade-offs between detection speed and model accuracy in hybrid approaches. With the current research, we cannot conclude whether a hybrid approach allows fast enough detection.

Robustness, while briefly discussed, also remains underexplored in the current literature. To assess the real-world applicability of a detection algorithm, it is crucial to evaluate the resilience of hybrid models against evasion techniques, adversarial attacks, and unforeseen challenges. While hybrid approaches are potentially more robust than single analysis-based approaches, no study has researched this to date.

None of the current hybrid approaches mention how they ensure their models' Scalability, Adaptation, and Flexibility. Nevertheless, a hybrid approach should do so to be suited for applicability in the real world.

Based on these observations, the applicability of hybrid approaches in automated phishing detection remains an open research question. The lack of consensus in the results from different studies motivates our hybrid framework. This framework combines multiple single-analysis-based models and consolidates their predictions using a stacking function. Our approach allows us to assess its applicability compared to individual models and provides a means to measure the increase in robustness, contributing to the ongoing exploration of hybrid models in this field.

4. Experimental setup

This section outlines the design and implementation of our experimental framework aimed at assessing the applicability of a hybrid approach to automated phishing detection. The framework encompasses various factors of applicability, including effectiveness, robustness, and speed of detection. The following sections detail our framework design, model selection process, and the proof of concept implementation.

4.1. Framework

From the literature section, we conclude that current studies on hybrid approaches often lack comprehensive discussions about their applicability. Instead, they tend to primarily focus on effectiveness, overlooking other crucial factors such as speed of detection, scalability, adaptation, flexibility, and robustness. Our objective is to construct a holistic framework that accounts for all factors of applicability. Among the six factors, effectiveness, robustness, and flexibility are mostly influenced by the architecture design. For instance, the choice of stacking

Table 2
Summary of the relationship between the factors of applicability and the design of the hybrid framework.

Applicability Factor	Relationship to Framework Design
Effectiveness	Influenced by stacking function choice; design impacts prediction quality.
Robustness	Enhanced by incorporating multiple input data types; design influences evasion resistance.
Flexibility	Design enables easy model replacement or enhancement, enhancing adaptability.
Speed of Detection	Dependent on the choice of individual models; framework design supports model selection for desired speed.
Scalability	Primarily influenced by individual models; framework architecture allows for scalability through model choices.
Adaptation	Predominantly driven by individual models; framework's adaptability supports model-based adaptation.

function has an impact on the effectiveness (Vecliuc et al. 2021), while robustness improves with the integration of multiple input data types (Venugopal et al. 2021). Additionally, ensuring flexibility necessitates the ease of implementing enhancements and extensions (Sahoo et al. 2017). In contrast, the framework's impact on the speed of detection, scalability, and adaptation appears limited compared to the influence of individual models. These factors mainly depend on the individual model designs (Sahoo et al. 2017). Consequently, fulfilling these criteria depends on the selection of models by the user.

The framework is arranged into three layers. The initial layer, the data layer, regards the types of data acquired from a website. These encompass elements such as the URL, HTML code, and other pertinent website attributes. For performance and robustness reasons, the data should maximize the website's information coverage to maximize the probability of finding useful information (Venugopal et al. 2021). This, in turn, enhances the framework's capability to detect potential phishing websites and reduces the likelihood of successful bypass attempts. However, it is worth noting that incorporating a greater variety of data inputs entails higher processing demands. Consequently, this may potentially lead to trade-offs in terms of flexibility, scalability, and speed of detection (Sahoo et al. 2017). Therefore, developers must thoughtfully balance these design principles while implementing their framework.

The second layer regards the individual models employed in the framework. The literature shows that different types of models perform better for different types of input features (Valiyaveedu et al. 2021). For this reason, we advocate for the integration of tailored models specific to each input feature rather than relying on a single generic model. This approach enhances the framework's flexibility as each model can be readily substituted with a superior alternative. Moreover, this strategy empowers customization to suit each end user's requirements. For instance, the choice of models significantly influences the speed of detection (Feng et al. 2020b). This flexibility accommodates users prioritizing the speed of detection and those emphasizing other applicability factors.

The framework's third layer combines the individual models' predictions using a stacking function. This function trains a meta-model that takes the predictions of the base models as inputs and produces the final prediction. Using a stacking function enhances the framework's flexibility by facilitating effortless replacements or improvements to its implementation. Moreover, the stacking function contributes to the framework's transparency, providing insight into its decision-making process (Do et al. 2022). A visual representation of the complete framework can be found in Fig. 1.

In conclusion, our hybrid framework strives to encompass a wider range of applicability factors as these are neglected by other hybrid approaches. By making deliberate design decisions, we impact effectiveness, robustness, and flexibility. Concurrently, the selection of individual models assumes a critical role in achieving the goals of speedy detection, scalability, and adaptability. This holistic approach acknowledges the intricate balance between design choices and model selection, ultimately shaping the framework's ability to cater to varying user needs. To illustrate the interplay between the six factors of applicability and the design of our hybrid framework, we present a summary in Table 2.

4.2. Proof of concept

To validate our framework, we create and test a proof of concept. This allows us to answer the research questions. We first elaborate on the scope of the proof of concept concerning the applicability of a hybrid approach. Next, we discuss the design choices for the proof of concept, which models we selected, and how we implemented them.

Since the framework allows us to incorporate all types of models, we first define the scope of our proof of concept. The applicability consists of six different factors that can result in a trade-off. For example, deep learning models tend to obtain a better performance but take a longer time for detection (Feng et al. 2020b). Since this is, to the best of our knowledge, the first study into the applicability of a hybrid approach, we scope our research down to three of the six factors: effectiveness, robustness, and speed of detection. Effectiveness is the factor mainly focused on by other studies. Furthermore, current hybrid approaches do not agree on the influence of a hybrid approach on effectiveness. The second factor we focus on is robustness. This tells us to what extent the proof of concept can withstand attackers' evasion techniques. Thereby, we can verify whether a hybrid approach is indeed a solution for more robust algorithms (Do et al. 2022). Finally, we evaluate the time required for detection, specifically examining the additional time taken by the hybrid framework compared to the individual models it comprises. We do not focus on the other factors of applicability for the proof of concept. These require a more extensive and qualitative-oriented analysis, which falls outside the scope of this study. This proof of concept aims to obtain initial findings regarding the applicability of a hybrid framework.

The first design choice for the proof of concept is which input features to incorporate. This choice depends on the availability and characteristics of models suitable for each type of input feature. Literature shows that most visual analysis-based models use a black- and whitelist approach (Abdelnabi et al. 2020; Dunlop et al. 2010; Khandelwal and Das 2022). To apply this, they use a dataset containing only a limited number of brands for the model to classify. As we want our proof of concept to be able to analyze each website on the Internet, we do not incorporate such an approach. Van Dooremaal et al. (2021) propose a visual-based approach but do not focus only on a limited number of brands. However, their method analyses phishing websites that are still online. Since phishing websites have a short lifespan of 21 days on average (Oest et al. 2020), this approach is unsuitable for most offline datasets. Therefore, we do not incorporate this or another visual analysis-based model into our proof of concept.

As we do not use visual features of a website as an input, there remain three other types of features for the proof of concept to analyze: the URL, HTML content, and HTML DOM tree structure. The URL and HTML content have shown to be effective in previous research (Valiyaveedu et al. 2021). The HTML DOM tree is a less explored topic. Nevertheless, it widens the information analyzable by the proof of concept. Although both the HTML content and HTML DOM tree structure originate from the HTML of a website, we assume they extract different insights from it. We base this assumption on how these models analyze the HTML code. The DOM tree analysis extracts the tree structure and tags from the HTML, leaving out the other content. Meanwhile, the content analysis analyses the complete HTML code but truncates it to a certain length to reduce the data size. Thereby, it loses information on the DOM tree structure. As a robustness check for this assumption, we compare the performances of the HTML content and HTML DOM tree model in the experiments section.

With our input features clarified, the next critical step is selecting appropriate models that align with the distinct website feature types. Given our aim of analyzing the URL, HTML content, and HTML DOM tree structure, we need three tailored models, each uniquely suited to their respective input features. For each model, we discuss the selection reasons and method of implementation. The outputs generated by these models serve as the input for our stacking function. In our experiments, we evaluate six distinct stacking functions within our proof of concept framework. This comprehensive assessment allows us to address our sub-question regarding the optimal stacking function for our hybrid framework. The experimental results guide us in determining the most suitable stacking function for our framework, enhancing its real-world applicability.

Model 1 - URL

The first model we implement in our prototype takes the URL as input. Such models obtain the highest scores within the fastest amount of time. Multiple studies claim an accuracy above 99% (Maneriker et al. 2021; Tang and Mahmoud 2021; Yang et al. 2021). However, these scores do depend on the datasets used. Therefore, no outspoken top-performing algorithm exists. Nevertheless, the URLnet algorithm stands as a benchmark (Le et al. 2018). We implement this model in our framework as it best represents the URL-based models.

The URLnet algorithm uses a combination of Character-based and Word-based embedding. This structure maintains the information contained in rare words while not hitting memory constraints. This is relevant since URLs contain high amounts of non-sensical terms. The algorithm's structure consists of the Character-level CNN and the Word-level CNN.

For the Character-level CNN, we convert the URL to a matrix representation. This data structure makes the URL analyzable as an image for the algorithm. To do so, we transpose each URL into a sequence of characters with a length of 200. We cut off URLs that are longer than 200, and we pad the ones that are shorter. Next, we embed each character in a 32-dimensional vector. Since we do this for all 200 characters, we obtain a 32 x 200 matrix. We pass this matrix through the convolutional layers, the main building block of a CNN. A convolutional layer consists of multiple filters. Each creates a different feature map, indicating the presence of detected features in the input matrix. Next, we add a Max-pooling layer. Max-pooling shrinks the size of the matrix while maintaining the essential features. We do so to save computational costs and avoid overfitting. Overfitting arises when the algorithm adjusts too much on the training data and fails to function well on the test data. After the Max-pooling layer, we put a fully connected layer with dropout. Dropout also helps prevent overfitting by randomly shutting off connections in the network. We concatenate the result of this with that of the Word-level embedding part.

The Word-level CNN is somewhat similar to Character-level CNN. The difference is that we have vector representations of words instead of characters. Since the different potential words in a URL are countless, this requires a different approach. First, we split the URL into a list of words. We do so based on the special characters ('/', '.' etc.). We remain the special characters '-' and '_' within the words. We use padding to give each list of words the same length of 200. Similar to character-level embedding, each vector representation is 32-dimensional. Accordingly, we express each URL as a 32 x 200 matrix. Since the algorithm has to store each different word, we can run into memory constraints. We replace all single-occurring words with an 'unknown' token to prevent this.

The single-occurring words, removed for word-level embedding, can still obtain information. Therefore, we add another part to the word-level embedding: a character-level embedding of rare words and special characters. This addition allows us to represent them by a fixed amount of characters. Hence, we maintain some information within these words while not facing memory constraints. We pad or truncate all words in the URL to a length of 20 characters. Each character gets a 32-dimensional vector representation. These steps result in a 32 x 20 matrix for each word. Next, we add those together for all 200 words in the URL to get a 200 x 32 matrix. We can add this to the matrix obtained from the word-level embedding since they have the same size. This leaves us with one word-level matrix containing information from the frequently occurring and unique words. We input this resulting matrix in a CNN identical to the one described for character-level embedding.

Finally, we concatenate the results from the character-level and word-level-based CNNs. We follow this with fully connected layers and a final layer, which gives a prediction. We use backward propagation to adjust the weights during the training phase. Table 3 shows the exact hyper-parameters we use for the model. We incorporate this for reproducibility purposes.

Table 3
Parameter settings for the URL-based model.

Parameter	Value
Length words	200
Length characters	200
Learning rate	0.001
Batch size	128

Table 4
Parameter settings for the HTML Content-based model.

Parameter	Value
Length words	200
Length characters	200
Learning rate	0.0015
Batch size	20
Dimensions	100
Number of words	400000
Number of characters	360

Model 2 - HTML content

We assess HTMLphish by Opara et al. (2020) to analyze the HTML content. This model is most suitable for HTML content analysis based on our literature review. Although Bilot et al. (2022) obtain better accuracy, we prefer HTMLphish over their model because their approach requires a crawler to extract features from the webpages recursively, making it incompatible with currently existing datasets.

The method employed by HTMLphish is similar to URLnet. It uses a CNN on both character and word-level embedding. To use this, we first transform the HTML code into a string using the Beautiful Soup library (Richardson 2022), a Python package for parsing HTML and XML documents. We consider special characters as separate tokens at the word level split, as these are common in HTML. For the character level split, we split on each character, for both approaches, we pad and truncate to a certain length. Although this step loses information, obtaining equal-length vectors for the embedding is required. Furthermore, it handles the capacity problem for long HTML files. We concatenate the character- and word-level embedding layers as the next step. In the resulting concatenated layer, we parse through the CNN. Table 4 shows the exact hyperparameter settings to reproduce this model.

Model 3 - HTML DOM tree

Our proof of concept also supports the HTML DOM tree as input. This data type contains information embedded in the structure of the HTML. The literature section discusses two single-analysis-based models that use this input type (Feng et al. 2020a; Ouyang and Zhang 2021). Ouyang and Zhang (2021) obtain the highest accuracy with their approach by combining an RNN with a GNN. Although GNNs seem a promising research direction, they show limitations in terms of appli-

Table 5
Parameter settings for the HTML DOM tree-based model.

Parameter	Value
Vector size	300
Learning rate	0.065

cability. These models consume significant amounts of memory, which goes at the expense of Scalability and Speed of detection Zhang et al. (2022). As we prioritized the practical utility and real-world applicability of our framework, we opted for a different approach that aligns more seamlessly with our research objectives. The method proposed by Feng et al. (2020a) uses Doc2Vec, a commonly used model (Le and Mikolov 2014). As this latter method is better applicable, we implement this approach for the DOM tree analysis (Table 5).

In this section, we devised the hybrid framework aimed at applicability. Furthermore, we created a proof of concept of the framework that contains three single analysis-based models. These analyze the URL, HTML content, and HTML DOM tree structure. In the next section, we assess the proof of concept on effectiveness, speed of detection, and robustness. Also, we compare different stacking functions for the proof of concept.

5. Experiment

This section presents the experiments performed to assess the applicability of the proof of concept. In particular, we aim to answer the following sub-research questions:

RQ1: How effective is a hybrid framework for automated phishing detection?

RQ2: What is the speed of detection of a hybrid framework to detect phishing in an automated system?

RQ3: To what degree is a hybrid framework for automated phishing detection robust to bypassing efforts?

RQ4: Which stacking function performs best in a hybrid framework for automated phishing detection?

By answering these questions, we comprehensively understand the hybrid framework's potential for automated phishing detection. Our experiments contribute to assessing the applicability of the proof of concept and shed light on the nuances of its effectiveness, detection speed, robustness, and different stacking functions. These insights are crucial for evaluating the practicality and reliability of the hybrid framework in real-world scenarios, ultimately contributing to the advancement of automated phishing detection methodologies.

We conducted our experiments on an Ubuntu 22.04.1 virtual machine boasting 24 GB of RAM. Our setup did not involve a GPU but focused on CPU performance. Moreover, the hardware employed for the experiments is relatively basic: An HP ZBook Power G8 with i7-11800H CPU operating at 2.3 GHz for our evaluations. As with Experiment 2, we split the data for the machine-learning-based model into a training and test set. We used relatively cheap hardware to test our experiments since we decided to implement a system that could run in a real-life environment and use easily affordable and available hardware.

Next, we present the dataset used for the experiment. Then, we present the setting of each experiment along with the results.

5.1. Dataset

For our experiments, we targeted a publicly available dataset that provides a complete representation of websites and is recent. This is because models trained on an outdated dataset can decrease performance when tested on more recent ones (Sánchez-Paniagua et al. 2022a). Among the existing datasets, the PILWD-134 K dataset (Sánchez-Paniagua et al. 2022b) includes the URL, HTML, screenshots, and metadata of phishing websites. Although this seems to be the most suitable dataset, it was unavailable when conducting this work. Another dataset that contains a comprehensive representation of the websites was proposed by Chiew et al. (2018). However, this dataset has two limitations. First, the data is rather old. The scraping took place between March and April 2016. Second, the URL data is not representative of the real world. The legitimate class contains only the second-level and top-level domains of the URL, while the phishing class also includes the scheme, subdomain, and sub-directories. The use of this dataset can introduce bias as the detection model can only check for the presence of the scheme in the URL. However, in real life, both phishing and legitimate websites contain the scheme. Therefore, this dataset is unsuitable for our study.

To the best of our knowledge, PILWD-134 K and the dataset presented by Chiew et al. (2018) are the only ones providing a comprehensive representation of websites. As these are unavailable or unsuitable, we require a different dataset that might contain fewer website features. The Phishing Website Dataset contains a recent collection of the URL and HTML of legitimate and phishing websites (Ariyadasa et al. 2021). Both these website characteristics show good predictability when analyzed. Furthermore, they allow for supervised learning without requiring brand labeling. The dataset contains 80,000 website samples, divided into 50,000 legitimate and 30,000 phishing, scraped from December 2020 to November 2021. The legitimate websites originate from the Ebbu2017 phishing dataset and the top Google search results for simple keywords (Sahingoz et al. 2019). The phishing websites come from PhishTank (PhishTank 2022), OpenPhish (OpenPhish 2022) and PhishRepo (Ariyadasa et al. 2022).

We download this dataset from the Mendeley Dataset repository (Ariyadasa et al. 2021). For training, we split them into a training (70%), validation (20%) and testing (10%) subset. Furthermore, we create three different representations, each containing 80,000 samples. The first one consists of the URLs. The other two contain the HTML code, of which one includes the complete content and the second only the DOM tree structure.

5.2. Experiments

5.2.1. Experiment 1: effectiveness

In the first experiment, we delve into a key factor of our research—evaluating the effectiveness of our hybrid framework for automated phishing detection. Within this study's scope, effectiveness is defined as the framework's capacity to distinguish phishing websites from their legitimate counterparts accurately. As this is the core task of a detection model, most research focuses solely on this factor. However, we want to stress that an effective approach does not directly mean it is applicable and, therefore, suitable for real-world use.

As different evaluation criteria for predictive performance exist, we determine the most suitable based on what they represent. Fig. 2 shows the four classes of prediction that the model can make. If predictions are correct, only True Positive (TPs) and True Negative (TNs) occur. Hence, the goal is to minimize False Positive (FPs) and False Negative (FNs). An FP prediction means an Internet user is warned not to trust a legitimate website. The danger of receiving frequent FPs is that users might question the system's reliability. The occurrence of an FN allows an attacker to phish successfully. Considering the consequences of phishing, we conclude that minimizing the FNs is more important than reducing the FPs.

Other metrics commonly used are (Alshingiti et al. 2023; Bu and Cho 2021a; Feng et al. 2018; Wang et al. 2019):

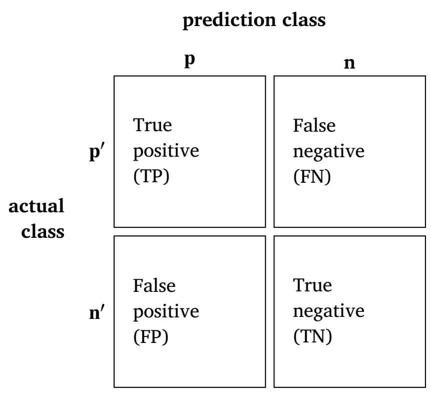

Fig. 2. Prediction outcomes of binary classification.

- Accuracy: The ratio of correct predictions over the total amount of predictions.
- Precision: The ratio of correctly classified phishing websites over the total number of predicted ones.
- Recall: The ratio of correctly classified phishing websites over the total amount of actual phishing websites.
- F1-score: The weighted average of Precision and Recall
- ROC curve: The Receiver Operating Characteristics curve shows the trade-off between the TPR and the FPR.
- ROC AUC: The Area Under the ROC Curve quantifies the ROC curve. It shows the ability to classify the correct label. This metric is used for the comparison of ROC curves.

Accuracy is typically used for classification purposes. However, it is not optimal in our case because we are more interested in the detection of phishing pages than legitimate ones. Therefore, F1-score best fits our needs, as it balances Precision and Recall.

To enable a meaningful performance comparison between our proof of concept and established models, we evaluated them alongside the three single analysis-based models we incorporated in the proof of concept. This deliberate choice allows us to assess their performance under consistent conditions, thereby mitigating the potential confounding effects of varying datasets. By doing so, we ensure a more robust and insightful analysis, as comparing our results directly with models from different studies could introduce extraneous variables stemming from dissimilar dataset characteristics, as highlighted by Sánchez-Paniagua et al. (2022a).

This experiment gives us the effectiveness of the individual models incorporated in the proof of concept. To assess the effectiveness of the proof of concept, we need to combine the predictions of the individual models. We do this using a stacking function (see Experiment 2).

Results Experiment 1
Table 6 shows the results obtained from this experiment. The first column indicates the three models incorporated into the model. These are indicated by the type of input data they process. The other columns report the score for each model for different metrics. The scores in the table show differences in performance. Based on the F1 score, the URL-based model shows the most effective with 93.82%. However, on Precision, the Content-based model outperforms the URL one. This indicates that the Content-based model has a higher tendency to classify websites as phishing. The DOM-based model falls behind on both metrics. Comparing this to the Accuracy, it shows a tendency to overclassify websites as legitimate.

5.2.2. *Experiment 2: stacking function*

In this experiment, we explore the capability of various stacking functions to combine the predictions of individual models. Stacking functions are crucial in improving the performance of our phishing detection proof of concept. In particular, we expect them to improve their effectiveness and robustness. In this experiment, we compare the effectiveness scores of the different stacking functions, as this is the core task of a phishing detection model.

Table 6
Performance of each individual model implemented in the proof of concept.

Model	Accuracy	Recall	Precision	F1-score	ROC AUC
URL	95.41%	94.51%	93.15%	**93.82%**	94.96%
Content	93.64%	88.57%	95.29%	91.81%	93.97%
DOM tree	90.30%	87.29%	86.70%	86.99%	89.58%

The task of the stacking function is to combine the predictions of the individual models incorporated. This approach offers a unique opportunity to harness the strengths of these individual models. However, the extent to which this approach proves effective and which stacking functions are most suitable remains a gap in the current literature. Therefore, our experiments aim to better understand the impact of various stacking functions on both effectiveness and robustness instead of aiming for mere effectiveness maximization. It is worth noting that our study does not venture into the development or inclusion of a novel stacking function for maximum effectiveness, as this falls outside the scope of our research. Nevertheless, we believe that this aspect of our approach introduces a valuable and innovative contribution to the field of automated phishing detection.

To assess the impact of the stacking function, we compare six different stacking functions, each one offering a distinct perspective on how to combine the predictions of individual models effectively. We make our selection based on several factors. First, we aim to evaluate our proof of concept within the established practices of phishing detection, leading us to include commonly used stacking functions. This approach offers insights into the performance of our method against existing techniques. Secondly, we deliberately incorporate stacking functions representing diverse methodologies to assess their varying impacts on effectiveness and robustness. This comprehensive approach encompasses both directly applicable stacking functions and machine learning-based ones, spanning a wide spectrum of techniques. Lastly, we prioritized practical applicability, emphasizing the need for stacking functions that enhance both effectiveness and real-world feasibility, ensuring their suitability for deployment in real-time phishing detection systems while considering computational efficiency and practicality. Based on these factors, we select the following stacking functions:

1. *Mean Predictions*: Mean predictions involve calculating the average of the predictions made by the individual models. The resulting value is then rounded to determine the final classification. This approach aims to find a consensus among the models by considering their collective wisdom.
2. *Majority Vote*: The majority vote is a straightforward stacking function where each individual model makes its prediction, rounding it to either 1 (indicating phishing) or 0 (indicating legitimate). The majority vote aggregates these predictions, and the outcome with the most votes becomes the final verdict. This approach gives equal influence to each model for every prediction.
3. *Most Certain Prediction*: In this approach, we rely on the model with the highest prediction confidence. Instead of an even influence distribution, this method gives all the decision-making power to the model with the highest certainty regarding the classification. This approach seeks to leverage the expertise of the model with the greatest confidence.
4. *Decision Tree*: We include the Decision Tree classifier due to its usage in a related study by Venugopal et al. (2021). While Decision Trees are known for their interpretability and the ability to handle complex feature interactions, they can also be prone to overfitting. By including this method, we aim to investigate its performance in our specific context and explore whether it offers advantages over other stacking functions.

Table 7
Performance of different stacking functions for the proof of concept. The results from Table 6 are added for comparison.

Function	Accuracy	Recall	Precision	F1-score	ROC AUC
URL	95.41%	94.51%	93.15%	93.82%	94.96%
Content	93.64%	88.57%	95.29%	91.81%	93.97%
DOM tree	90.30%	87.29%	86.70%	86.99%	89.58%
Mean	97.33%	96.80%	96.02%	96.41%	97.06%
Vote	96.81%	95.27%	96.26%	95.76%	96.70%
Certain	97.21%	97.73%	94.75%	96.22%	96.72%
Decision tree	95.62%	94.89%	93.95%	93.84%	94.92%
Random Forest	97.44%	98.09%	94.62%	96.50%	96.93%
Logistic regression	97.44%	96.32%	96.81%	**96.56%**	97.31%

5. *Random Forest*: The Random Forest stacking function is an ensemble method that combines multiple decision trees. It has demonstrated effectiveness in similar studies, such as the work by Vecliuc et al. (2021). Random Forests are known for reducing overfitting, improving overall classification accuracy, and enhancing model robustness. We included this stacking function to assess whether ensemble methods can provide significant performance gains in our proof of concept.
6. *Logistic Regression*: Logistic Regression is a well-established machine learning algorithm known for its effectiveness in binary outcome prediction. We included this stacking function because it aligns well with our primary goal of distinguishing phishing websites from legitimate ones. Its simplicity and interpretability make it an attractive choice for real-world applications where model transparency is essential.

The Decision Tree, Random Forest, and Logistic regression stacking functions use machine learning. These require us first to train the algorithms. To do so, we first split the list of predictions outputted by the individual models (8,000 websites) into a train and test split (80%-20%). This allows the stacking function to understand which value they should give to the predictions of the different models.

In this experiment, we compare the effectiveness of different stacking functions. We do so by testing the proof of concept six times, each time with a different stacking function. We test the performance of the different stacking functions by the metrics determined for Experiment 1. Consequently, we assess the F1 score as the most critical metric. Each test result shows how the proof of concept performs when we use the stacking function to combine the predictions of the individual models. We assess the robustness of the different stacking functions in Experiment 4.

Results Experiment 2
Table 7 shows the performance of the proof concept for each of the stacking functions. The three upper functions represent the directly applicable ones, which use 8000 websites to test their performance. The bottom three functions are the machine learning ones. These used 80% of the websites for training. Their results in the table originate from the remaining 20% of websites.

The proof of concept performs better with each stacking function than the best individual model (URL). Although Logistic regression performs best, no outperformer exists. Only the Decision tree turns out to perform slightly less.

5.2.3. *Experiment 3: time*

We measure the training and test time of each model on the dataset. The training time of the proof of concept is equal to the sum of the training time of the incorporated models. This is the same for the testing time plus the time it takes the concatenation function.

Results Experiment 3
The training and testing times are presented in Table 8. We see two clear findings. First, the incorporated models differ greatly in training

Table 8
Time it took for the individual models to train and test the data. We did not include the time of the different stacking functions, as each of them would be below 0.

Function	Total time (s)		Average time (ms)	
	Train	Testing	Training	Testing
URL	2652	28	36.8	3.5
Content	19508	37	270.9	4.6
DOM tree	32012	118	444.6	14.8

and testing time. For instance, the URL model demonstrated rapid training, while the DOM tree model exhibited relatively slower performance. The average training time per sample spanned from 36.8 milliseconds to 444.6 milliseconds. The second finding is regarding the speed of the different stacking functions. Contrary to the results observed for the models, where training and testing times varied significantly, both training and testing times for the stacking functions were consistently rapid, requiring less than a second in total and less than a microsecond on average. The overall training process, including the stacking function, amounted to 15 hours, 2 minutes, and 52 seconds, translating to a mere 0.75 seconds per website. In contrast, testing occurred much faster, with an average time of 22.9 milliseconds per sample. These results emphasize that the stacking functions introduce minimal computational overhead, reinforcing their applicability for real-world deployment.

5.2.4. *Experiment 4: robustness*

By testing the robustness, we aim to measure the impact of an attacker bypassing one of the individual models on the overall detection abilities. To measure this, we simulate a phisher exploiting a flaw in one of the individual models. This results in the bypassed individual model giving false predictions. To be robust, the other individual models should be able to compensate for this flaw. Suppose a phisher can circumvent the complete proof of concept by exploiting only one of the individual models incorporated. In that case, we can conclude that the proof of concept is not robust.

AlEroud and Karabatis (2020) show an example of bypassing a URL-based model with GANs. If a cybercriminal manages to do so, it wants the model to label a phishing website (value = 1) as legitimate (value = 0). It is clear how to simulate this for voting. We take the individual model for which we simulate a bypassing and change all its classifications of phishing to legitimate. This change results in the individual model classifying each website as legitimate.

It is more complex to simulate a bypass when we use probability predictions. To the best of our knowledge, no previous literature exists on this topic, so we reason on what consequence a bypass may have. Bypassing an individual model means it assesses a phishing website as benign. Thus, the model gives a probability prediction below the threshold of 0.5, where it used to give one above it. Which exact probability it becomes is unknown, as this depends on the convincingness of the bypassing method. However, we can determine for which probability

Table 9
F1 score of the proof of concept for each type of stacking model while bypassed. Each column represents a bypassing for that specific data type model. The bypass only just managed to beat the threshold. Therefore, each phishing website has a phishing prediction of 0.5 or lower.

Stacking Function	URL	Content	DOM tree	No bypass
Mean	92.57%	95.48%	94.79%	96.41%
Vote	89.84%	89.19%	93.80%	95.76%
Certain	**92.67%**	96.06%	95.17%	96.22%
Decision tree	88.16%	91.69%	91.27%	93.84%
Random Forest	88.97%	95.70%	94.39%	96.50%
Logistic regression	92.60%	**96.13%**	**95.69%**	**96.56%**

Table 10
F1 score of the proof of concept for each type of stacking model while bypassed. Each column represents a bypassing for that specific data type model. The phisher managed to bypass the model perfectly. Therefore, each phishing website has a phishing prediction of 0.0. Meaning a full confidence that the website is legitimate.

Stacking Function	URL	Content	DOM tree	No bypass
Mean	78.54%	86.32%	88.86%	96.41%
Vote	**89.84%**	89.19%	**93.80%**	95.76%
Certain	0.00%	0.00%	0.00%	96.22%
Decision tree	17.66%	77.22%	0.63%	93.84%
Random Forest	43.26%	82.90%	0.00%	96.50%
Logistic regression	79.96%	**92.97%**	88.93%	**96.56%**

prediction the phisher will strive. The phisher aims to bypass the model to start exploiting the flaw. Although they might further optimize the bypassing method, we do not expect them to do so. The phisher does not know how the model evaluates and improves its decision-making methods. Therefore, it would be risky to optimize the bypassing strategy instead of exploiting the opportunity. Therefore, a minimal bypass suffices.

We simulate the bypass by changing the probability prediction of a phishing website formerly recognized by the individual model ($p > 0.5$) to the probability just below the threshold of phishing ($p = 0.5$). As phishers will start exploiting this bypass immediately, we apply this change to all data points in the above category. We leave phishing websites that could already fool the individual model ($p < 0.5$) untouched. We execute this simulation for each of the three individual model predictions. Thus, three separate proof of concepts are performed, each simulating a bypass for one of the individual models. Furthermore, we test each proof of concept with the different stacking functions. To assess the performance, we compare the F1 scores. If the decrease in F1 score is insignificant, we conclude the proof of concept is robust. A single analysis-based model would obtain an F-1 score of 0.0% in case of a bypass, as they would not be able to identify a single phishing website correctly. This makes it hard to compare when a performance decrease is significant.

We execute the above steps a second time, but we simulate three perfect bypasses of the individual models this time. By perfect, we mean that the bypassing method fully convinces the model that the phishing website is legitimate. We simulate this by changing the probability prediction for each phishing website to completely legitimate ($p = 0$). This also includes the phishing websites previously able to deceive the individual model but less convincingly ($0 < p < 0.5$). Although this situation is improbable in practice, it shows how the proof of concept would perform in the most hostile case.

Results Experiment 4
The performance of the proof of concept when one of the individual models is bypassed is shown in Table 9. Each column reports the individual model for which we simulated a bypass. The fourth column, “No bypass”, shows the F1-score when no bypass occurs. These values are the same as in Table 7 and are shown for comparative purposes. The first observation is that the performance decreases at the bypass for every stacking function. Looking at the highest F1-score for each column, the column of the URL model is the lowest. This means that the impact of bypassing is most prominent when we simulate a bypassing for the URL model. As we saw in Table 6, this is the best-performing individual model in Experiment 1. Looking at the lowest impact of a bypass, Table 7 shows we do not obtain this in case of a bypass of the worst performing individual model. The F1-score in the Content column is higher than in the DOM tree one. However, this only is a difference of 0.44%, which shows that the impact on the F1 score in case of bypassing the worst-performing individual model is much lower than for the best-performing one.

The Logistic regression remains the best-performing model in case of a bypass. This method works similarly to the mean function by taking the prediction of each model. However, it applies a different factor to each prediction, which results from the training process. As both the mean and logistic regression functions keep performing well, it shows the importance of looking at the confidence of the predictions.

This could also declare why the Decision tree remains the worst-performing one. As this stacking function makes use of thresholds, it indicates the vulnerabilities of these thresholds to bypassing. Although some models perform much worse when the URL model gets bypassed, no extreme situations arise. Furthermore, some stacking functions maintain good performance on each bypassing. The average decrease in F1 score is 3.40%.

We also simulated a full bypassing of the individual models. Table 10 shows this impact on the proof of concept’s performance. We can immediately see this is large. The certain-based stacking function is not able to make accurate predictions anymore. Since this full bypass makes the model fully convinced that the phishing websites are benign, the other models appear unable to compensate for this. A similar situation arises for the Decision tree and Random Forest stacking functions. In particular, in the case of a full bypass for the DOM tree model. Apparently, this model serves as an important parameter for these functions and their classification.

On the other hand, the mean, voting, and Logistic regression functions maintain some performance. The voting performs equally to the more realistic bypass simulation. Since it works with a vote of the bypassed individual model, it does not matter to which extent it gets convinced. As the other stacking functions do receive hinder from this, the voting function switches from one of the worst performing functions to the best one. Furthermore, it remains to function adequately. For the mean-based function, the impact of the bypass depends on the predictive performance of the individual model. If this is higher, the impact seems more significant. The Logistic regression performs relatively the same as with a more realistic bypass. Merely, the impact of the bypass increases.

5.3. Comparison to the state of the art

To qualitatively evaluate the contribution of our research, we opted to benchmark it against the top-performing hybrid approach. To the best of our knowledge, the current state-of-the-art model in this regard is the Web2Vec proposed by Feng et al. (2020b). We used a training set (70%), test set (20%), and validation set (10%) to perform our comparison experiment. The model was trained with the training and validated with a separate validation set. Lastly, it has been evaluated on a third and separate test set.

Table 11 reports the results of our experimentation using Web2Vec. The table shows the effectiveness and speed of detection of Web2Vec and our hybrid framework. Regarding the effectiveness, we observe that our approach yields a higher accuracy compared to the results from Feng et al. (2020b). Based on this, we can conclude that we perform better than the current state-of-the-art approaches. Moreover,

Table 11
Comparative effectiveness and speed of detection between our hybrid framework and the current state-of-the-art approach.

Approach	Effectiveness		Training Time		Testing Time	
	Accuracy (%)	F1-Score (%)	Total (s)	Average (ms)	Total (s)	Average (ms)
Our Hybrid Framework	**97.44**	96.56	**54172**	**752.39**	**183**	**22.88**
Feng et al. (2020b)	97.21	**97.75**	115795	1608.26	449.58	56.19

from Table 11, we highlight the remarkable difference in the detection speed. Our comparison experiment shows that our approach is more than twice as fast in both training and testing times. Therefore, our approach requires less computational time and is better suited for real-time detection.

6. Discussion

The results of the experiments help us determine how suitable a hybrid framework for automated phishing detection is in practice, thereby answering our sub-questions and research questions. We tested the proof of concept's effectiveness, time, and robustness for different stacking functions. To assess these results, we compare them to the results of the single analysis-based models incorporated into the proof of concept. As these are benchmark models tested on the same dataset, we expect this to give the best comparison between the hybrid approach and currently used models. This limits the risk of having other factors influence the results. Such a situation could arise when we compare the hybrid approach to single-analysis-based models tested in different studies, as a difference in results could be explained by the datasets (Sánchez-Paniagua et al. 2022a). We assess how this relates to and contributes to current scientific work on this topic. Furthermore, we discuss the limitations of our current study and how further research can extend our work.

The first part of this section discusses the results of the different experiments. The second part infers what these results tell about the hybrid approach in general. The last part discourses the societal impact of the study.

6.1. Proof of concept

Effectiveness
Our first sub-question is about the effectiveness of a hybrid approach. The results show that the proof of concept, with an accuracy of 97.44% and an F1 score of 96.56%, excels in distinguishing phishing websites from benign ones, outperforming each individual model. This indicates that a hybrid approach leads to better effectiveness than a single analysis-based model. Notably, our results align with the recommendations from Do et al. (2022), who advocated for exploring hybrid approaches. However, this expectation is not uniformly confirmed by other studies employing hybrid approaches that incorporate stacking functions, such as Vecliuc et al. (2021) and Venugopal et al. (2021). Their reported outcomes reveal equal or even reduced effectiveness of the hybrid approach compared to the individual models they integrate. We expect two factors to influence these contradictory results.

First of all, these studies show a larger variation in test score accuracy between the individual incorporated models. In particular, Venugopal et al. (2021) show accuracies that vary between 73.8% and 99.98%. This gap might be too big for the weaker model to add any value to the stronger model. A second reason for contradictory findings by these studies and ours is the dataset employed. These studies test the individual models on different datasets than the hybrid combinations. In contrast, our approach tested the proof of concept on the same dataset as the individual models. This approach enhances the realism of our findings as it ensures a direct comparison of performance in identical conditions. Additionally, our finding aligns with the study conducted by Feng et al. (2020b). They also conclude that a hybrid approach leads to higher effectiveness.

Speed of detection
Phishing detection solution should allow for real-time detection of phishing websites. Therefore, the detection speed is a critical factor in determining their practicable applicability. Feng et al. (2020b) suggest that the speed of detection depends on the type of model incorporated into the framework but do not provide an evaluation to support their claim. In this work, we empirically investigated the detection speed of our hybrid framework and the impact of employing a stacking function. To this end, we measured the training and test time of each implemented model and the different stacking functions. The results show that the time it takes the stacking function is negligible compared to that of the individual models. Therefore, we can deduce that a hybrid framework can achieve a high detection speed if it incorporates fast models. Sahoo et al. (2017) claim that fast detection models can classify a website in the order of milliseconds. Our results on the test data show that our proof of concept can indeed do this on average in ∼22 ms, thus making our approach suitable for real-time detection.

Robustness
The third factor of applicability we assess with our proof of concept is robustness. Specifically, we examine how a hybrid framework handles a bypass attempt on one of its integrated models, where the model mistakenly identifies a phishing website as legitimate. If not mitigated, such a situation could result in users trusting a phishing website. To explore this, we conducted simulations of two degrees of bypassing.

The first simulation represents a scenario in which a phishing attacker attempts to deceive the model with minimal effort, making it a more realistic representation. Our results indicate that this level of bypassing results in only a minor performance decrease for the entire architecture. The average drop in the F1 score was measured at 3.40%. In contrast, a single-analysis-based model would drop to an F1 score of 0% under similar circumstances. This finding suggests that a hybrid approach can effectively handle the impact of a bypass and can be deemed robust.

To further evaluate the proof of concept's ability to withstand a bypass, we conducted a second bypass simulation in which the bypassed individual model consistently classified all phishing websites as legitimate with full certainty. While this scenario is unlikely to occur in reality, it represents the worst-case scenario for the impact of a bypass on the overall detection algorithm. Our results show the importance of selecting a stacking function based on the importance of different applicability factors. While voting is not the most effective stacking function, it is the most robust. Such a trade-off depends on the requirements of the end-user.

With certain stacking functions, the proof of concept could still achieve a better F1 score than the least effective model incorporated, even in such an extreme bypass scenario. Additionally, it managed to maintain sensible classifications with some other stacking functions, although these did exhibit a substantial decrease in the F1 score. This implies that, in practice, the proof of concept would not completely fail its task when faced with a bypass.

While these findings align with the expectations of Do et al. (2022), they also provide insights for the development of more robust models. However, several caveats should be considered. First, our bypass simulations are theoretical and do not replicate a real-life hostile environment. Thus, the extent of realism in our simulation remains uncertain. Second, in practice, the appearance of a bypass may vary as hackers continually seek new methods, as concluded by Al Halaseh and Alqatawna (2016). A phisher could even attempt to bypass two individual models simultaneously. Our tests encompass scenarios where the phisher attempts to bypass or completely fool one of the models, whereas real-life situations often fall somewhere in between. Nevertheless, our belief is that assessing the framework under worst-case scenarios is essential, and our full bypass simulation provides valuable insights.

This comprehensive evaluation of robustness enhances our understanding of the hybrid approach's capabilities and limitations, paving the way for future research to develop even more resilient models. Moreover, it is worth mentioning that the work from Feng et al. (2020b) did not provide any experimentation on the robustness of the model. To the best of our knowledge, our study is the first research to test it empirically.

Stacking function

In exploring hybrid approaches, we recognized the importance of assessing the performance of different stacking functions concerning applicability. While much attention has traditionally been placed on their effectiveness, our study delves into the broader spectrum of applicability considerations.

Regarding effectiveness, our findings reveal that Logistic regression stands out as the top-performing stacking function. This observation can be attributed to its capacity to capture correlations between individual model predictions, enabling more effective utilization of their outputs. Notably, this aligns with the reviewer's point regarding the potential advantages of Logistic regression. Notably, most other stacking functions yielded nearly equivalent F1 scores. Interestingly, the Decision tree, employed by Venugopal et al. (2021), also performed comparably in our analysis. However, it is important to emphasize that their study did not provide a comparative assessment of multiple stacking functions or elaborate on the rationale behind their choice.

The applicability landscape shifts when considering robustness, a key facet of our study. Some stacking functions exhibited limitations in making reasonable classifications during the second-degree bypass simulation, indicating a lack of robustness. Consequently, we would not recommend incorporating these less robust stacking functions into a hybrid approach aimed at optimal applicability. Here, we encounter a trade-off between effectiveness and robustness. While the mean and Logistic regression stacking functions excel in effectiveness, voting emerges as the more robust choice. The selection between these two depends on the user's priorities and requirements. If maximizing accuracy and F1 score is paramount, the mean or Logistic regression may be preferred. Conversely, if maintaining a high degree of robustness, especially in the face of challenging scenarios like bypass attempts, is a priority, voting becomes the preferred option.

In summary, our stacking function analysis underscores the need to strike a balance between effectiveness and robustness when designing a hybrid approach. The choice should be driven by the specific objectives and preferences of the end-user, ensuring that the system can deliver optimal performance while remaining resilient under adverse conditions.

Approach

Our current approach fills an important gap in the literature by emphasizing applicability, robustness, and real-world viability. While we acknowledge the potential for further research avenues, we believe that our study's comprehensive evaluation provides valuable insights for the practical deployment of phishing detection systems.

Our approach offers several distinct advantages contributing to its value in the problem domain. Firstly, we focus not only on the individual models but also on their combined applicability. This perspective allows us to assess the effectiveness, speed, and robustness of the entire hybrid framework, providing a holistic view of its real-world utility. By emphasizing the broader spectrum of applicability, we move beyond the traditional single-focus evaluation of model effectiveness. Moreover, our study explicitly explores the robustness of the hybrid framework, demonstrating its resilience to bypass attempts, a crucial factor of real-world applicability. This adds a layer of sophistication to our approach, as it considers potential threats and adversarial scenarios.

Our research focused on investigating various stacking functions to construct a practical and applicable hybrid framework for phishing detection, driven by our aim to emphasize real-world utility and ease of adoption. Stacking provides an effective means of combining model outputs within our framework, aligning with our goal of obtaining initial insights into the applicability of such a hybrid approach. This pragmatic approach ensures our framework's deployability in practical settings and helps us understand the impact of different stacking functions on applicability. While advanced techniques like Multiview Learning and dynamic aggregation functions hold the potential to enhance the framework's effectiveness and robustness (Li et al. 2018), we leave this avenue open for future research, recognizing the potential for further exploration in enhancing technical aspects. Our current study focused on building a foundational framework ready to address the urgent issue of phishing detection.

6.2. Practical implications

Our study on the hybrid approach for automated phishing detection carries practical significance, going further than traditional effectiveness evaluations. The implications of our findings extend beyond academia, offering tangible benefits for users, organizations, and the broader cybersecurity landscape.

Strengthened Cybersecurity Measures: At the core of our findings lies an enhancement in the accuracy and speed of phishing detection through our proposed hybrid framework. This translates to strengthened cybersecurity measures, a critical need as phishing attacks grow in sophistication and frequency. For this reason, organizations and users can trust that the automated system is more adept at accurately identifying and blocking phishing attempts, reducing the risk of successful attacks.

Adaptability to Real-world Dynamics: Our emphasis on real-world applicability ensures that the hybrid framework is not merely a theoretical construct but a practical solution tailored for different dynamic environments. By evaluating the framework's detection speed and robustness against bypassing attempts, our study provides a pragmatic assessment of its adaptability. This adaptability is crucial, as it enables the deployment of our proof of concept framework with the assurance that it has been rigorously tested and optimized for the complexities of real-world phishing scenarios.

Guidance for Decision-makers: Our study offers valuable guidance for decision-makers by showcasing the robustness of the hybrid framework against bypass attempts. This resilience instills confidence in users and organizations, ensuring the system remains effective even when confronted with sophisticated attempts to deceive it. Making informed decisions about stacking functions further empowers decision-makers, allowing them to align the system's priorities with their own. This could be maximizing accuracy or prioritizing robustness.

Seedbed for Future Developments: Our experiments provide more than just a proof of concept; they serve as a seedbed for future developments in phishing detection systems. The demonstrated effectiveness of the hybrid framework, outperforming individual models, serves as a blueprint for developing new and more reliable phishing detection systems. The hybrid approach, with its combination of models and stacking functions, opens avenues for innovation and exploration in the ever-evolving landscape of cybersecurity.

Efficiency in Resource Utilization: Compared to the current state-of-the-art models, our hybrid framework achieves higher accuracy within a significantly reduced computational time. This efficiency in resource utilization is not just a theoretical advantage but a practical one. It positions our approach as a suitable solution for phishing detection, aligning with the need for timely and accurate threat identification in the face of evolving cyber threats.

In conclusion, the practical implications of our study and related results suggest that our proposed hybrid framework provides tangible benefits for users and organizations seeking to enhance their cybersecurity. The findings of our experiments offer practical insights that can guide the selection, deployment, and ongoing improvement of automated phishing detection systems.

7. Conclusion

In conclusion, our research fills critical gaps in the field of phishing detection by shifting the focus from effectiveness alone towards real-world applicability and robustness. While effectiveness remains pivotal, we recognize the need for a more holistic evaluation of phishing detection systems, considering multiple dimensions of applicability.

Our study introduces a comprehensive assessment framework, evaluating the effectiveness, speed of detection, and robustness of a hybrid approach. Compared to the current state-of-the-art approach (Table 11), our framework obtains a higher accuracy within half of the computational time. Additionally, we demonstrated the superiority of a hybrid framework over single-analysis-based models, achieving higher accuracy and F1 scores in distinguishing phishing websites when combining them. Furthermore, our research systematically examined robustness, simulating bypass scenarios and showcasing the system's resilience against adversarial challenges. Notably, our study is the first to empirically test robustness, underscoring its distinctive contribution to the field of phishing detection.

Lastly, on top of the technical outcomes of our proposed approach, we would like to underscore the implications of our research within both societal and scientific contexts. In Section 2.1, we already introduced some key contributions and societal impact. However, we want to insist more on the importance of our research and findings in fighting phishing. Phishing relies on human error, and according to IBM (2023), phishing is one of the top attack vectors. Our thorough literature review in Section 3, revealed that most of the existing solutions for phishing detection are URL-based, rendering them susceptible to evasion through Generative Adversarial Networks (GANs). Consequently, our research aims to explore and test innovative hybrid models for phishing detection with the goal of enhancing both reliability and robustness. Our proposed approach has demonstrated to be reliable with an accuracy of ~97%. However, at the same time, our model exhibits robustness against possible potential evasion strategies, as discussed in Section 5.2.4. In a world where artificial intelligence continually advances, we think that the *Robustness* of our proposed approach assumes a pivotal role as a distinctive asset in safeguarding users against phishing attacks. Moreover, the resilience of our model represents a crucial and valuable contribution to user protection in the evolving landscape of cybersecurity.

8. Limitations and future work

We acknowledge that the findings presented in our work stem from the analysis of a single proof of concept. While this specific implementation demonstrates applicability, it does not guarantee that the use of other individual models yields similar results. In a further study, we want to test an implementation of the framework with more models. This would allow us to test and compare different compositions of the hybrid approach. On implementing more models, we could also simulate bypassing multiple individual models at a time. For this further study, multiple models exist that could add value when included in the design (Abdelnabi et al. 2020; Bilot et al. 2022; Sánchez-Paniagua et al. 2022b).

Looking ahead, our findings pave the way for further research to explore other dimensions of applicability, such as flexibility, adaptability, and scalability of a system. In principle, these system requirements are enhanced by a modular architecture, making it easier to swap models and adjust designs to different environments and needs. We propose to test the framework over a longer period of time, with additional models, simulating complex bypass scenarios, and incorporating various existing models to enhance the hybrid design. We expect that our contribution extends beyond addressing current gaps, offering practical insights for the future development and deployment of robust phishing detection systems.

As phishing attacks continue to evolve, our multifaceted approach to evaluating and improving detection systems becomes increasingly relevant. By embracing a broader perspective of applicability, we are better equipped to counter emerging threats, reinforcing the importance of ongoing research in this critical cybersecurity domain.

CRediT authorship contribution statement

R.J. van Geest: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Writing – original draft. **G. Cascavilla:** Conceptualization, Data curation, Supervision, Validation, Writing – original draft, Writing – review & editing. **J. Hulstijn:** Conceptualization, Supervision, Validation, Writing – review & editing. **N. Zannone:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abbate, P., 2022. Internet crime report 2021. In: Federal Bureau of Investigation. bit.ly/CrimeReport2021.

Abdelnabi, S., Krombholz, K., Fritz, M., 2020. Visualphishnet: zero-day phishing website detection by visual similarity. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1681–1698.

Afroz, S., Greenstadt, R., 2011. Phishzoo: detecting phishing websites by looking at them. In: 2011 IEEE Fifth International Conference on Semantic Computing, pp. 368–375.

Al Halaseh, R., Alqatawna, J., 2016. Analyzing cybercrimes strategies: the case of phishing attack. In: 2016 Cybersecurity and Cyberforensics Conference (CCC). IEEE, pp. 82–88.

Al-Ahmadi, S., Alotaibi, A., Alsaleh, O., 2022. Pdgan: phishing detection with generative adversarial networks. IEEE Access 10, 42459–42468.

AlEroud, A., Karabatis, G., 2020. Bypassing detection of url-based phishing attacks using generative adversarial deep neural networks. In: Proceedings of the Sixth International Workshop on Security and Privacy Analytics, pp. 53–60.

Alhogail, A., Alsabih, A., 2021. Applying machine learning and natural language processing to detect phishing email. Comput. Secur. 110, 102414.

Aljofey, A., Jiang, Q., Qu, Q., Huang, M., Niyigena, J.-P., 2020. An effective phishing detection model based on character level convolutional neural network from url. Electronics 9 (9).

Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., Wang, Y., 2022. An effective detection approach for phishing websites using url and html features. Sci. Rep. 12 (1), 8842.

Allodi, L., Chotza, T., Panina, E., Zannone, N., 2020. The need for new antiphishing measures against spear-phishing attacks. IEEE Secur. Priv. 18 (2), 23–34.

Almousa, M., Furst, R., Anwar, M., 2022. Characterizing coding style of phishing websites using machine learning techniques. In: 2022 Fourth International Conference on Transdisciplinary AI (TransAI), pp. 101–105.

Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Van Essen, B.C., Awwal, A.A., Asari, V.K., 2019. A state-of-the-art survey on deep learning theory and architectures. Electronics 8 (3), 292.

Alshingiti, Z., Alaqel, R., Al-Muhtadi, J., Haq, Q.E.U., Saleem, K., Faheem, M.H., 2023. A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN. Electronics 12 (1).

Ariyadasa, Subhash, Fernando, Shantha, Fernando, Subha, 2021. Phishing Websites Dataset.

Ariyadasa, Subhash, Fernando, Shantha, Fernando, Subha, 2022. Phishrepo-dataset.

Bilot, T., Geis, G., Hammi, B., 2022. Phishgnn: a phishing website detection framework using graph neural networks.

Bu, S.-J., Cho, S.-B., 2021a. Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing url detection. Electronics 10 (12), 1492.

Bu, S.-J., Cho, S.-B., 2021b. Integrating deep learning with first-order logic programmed constraints for zero-day phishing attack detection. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 2685–2689.

Bu, S.-J., Kim, H.-J., 2022. Optimized url feature selection based on genetic-algorithm-embedded deep learning for phishing website detection. Electronics 11 (7), 1090.

Cao, Y., Han, W., Le, Y., 2008. Anti-phishing based on automated individual white-list. In: Proceedings of the 4th ACM Workshop on Digital Identity Management, pp. 51–60.

Chiew, K.L., Chang, E.H., Tan, C.L., Abdullah, J., Yong, K.S.C., 2018. Building standard offline anti-phishing dataset for benchmarking. Int. J. Eng. Technol. 7 (4.31), 7–14.

Chinnasamy, P., Kumaresan, N., Selvaraj, R., Dhanasekaran, S., Ramprathap, K., Boddu, S., 2022. An efficient phishing attack detection using machine learning algorithms. In: 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), pp. 1–6.

Do, N.Q., Selamat, A., Krejcar, O., Herrera-Viedma, E., Fujita, H., 2022. Deep learning for phishing detection: taxonomy, current challenges and future directions. IEEE Access.

Dunlop, M., Groat, S., Shelly, D., 2010. Goldphish: using images for content-based phishing analysis. In: 2010 Fifth International Conference on Internet Monitoring and Protection. IEEE, pp. 123–128.

Dutta, A.K., 2021. Detecting phishing websites using machine learning technique. PLoS ONE 16 (10), e0258361.

Ejaz, A., Mian, A.N., Manzoor, S., 2023. Life-long phishing attack detection using continual learning. Sci. Rep. 13 (1), 11488.

Elsadig, M., Ibrahim, A.O., Basheer, S., Alohali, M.A., Alshunaifi, S., Alqahtani, H., Alharbi, N., Nagmeldin, W., 2022. Intelligent deep machine learning cyber phishing url detection based on bert features extraction. Electronics 11 (22).

Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., Wang, J., 2018. The application of a novel neural network in the detection of phishing websites. J. Ambient Intell. Humaniz. Comput., 1–15.

Feng, J., Zhang, Y., Qiao, Y., 2020a. A detection method for phishing web page using dom-based doc2vec model. J. Comput. Inf. Technol. 28, 19–31.

Feng, J., Zou, L., Ye, O., Han, J., 2020b. Web2vec: phishing webpage detection method based on multidimensional features driven by deep learning. IEEE Access 8, 221214–221224.

Hou, Y.-T., Chang, Y., Chen, T., Laih, C.-S., Chen, C.-M., 2010. Malicious web content detection by machine learning. Expert Syst. Appl. 37 (1), 55–60.

IBM, 2023. Cost of a data breach report. https://www.ibm.com/downloads/cas/E3G5JMBP.

Kexin, X., Liang, B., Rai, A., Chan, A., 2021. URL classification with deep learning.

Khandelwal, S., Das, R., 2022. Phishing detection using computer vision. In: Computer Networks and Inventive Communication Technologies. Springer, pp. 113–130.

Kim, T., Park, N., Hong, J., Kim, S.-W., 2022. Phishing url detection: a network-based approach robust to evasion. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22. Association for Computing Machinery, pp. 1769–1782.

Le, H., Pham, Q., Sahoo, D., Hoi, S.C., 2018. Urlnet: learning a url representation with deep learning for malicious url detection, arXiv preprint arXiv:1802.03162.

Le, Q., Mikolov, T., 2014. Distributed representations of sentences and documents. In: International Conference on Machine Learning. PMLR, pp. 1188–1196.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Lee, J., Ye, P., Liu, R., Divakaran, D.M., Chan, M.C., 2020. Building robust phishing detection system: an empirical analysis. In: NDSS MADWeb.

Li, Y., Yang, M., Zhang, Z., 2018. A survey of multi-view representation learning. IEEE Trans. Knowl. Data Eng. 31 (10), 1863–1883.

Lin, Y., Liu, R., Divakaran, D.M., Ng, J.Y., Chan, Q.Z., Lu, Y., Si, Y., Zhang, F., Dong, J.S., 2021. Phishpedia: a hybrid deep learning based approach to visually identify phishing webpages. In: USENIX Security Symposium.

Liu, E., 2021. Phishing webpage classification method based on joint features. In: 2021 3rd International Conference on Applied Machine Learning (ICAML), pp. 24–27.

Liu, R., Lin, Y., Yang, X., Ng, S.H., Divakaran, D.M., Dong, J.S., 2022. Inferring phishing intention via webpage appearance and dynamics: a deep vision based approach. In: 31st USENIX Security Symposium (USENIX Security 22). USENIX Association, pp. 1633–1650.

Maneriker, P., Stokes, J.W., Lazo, E.G., Carutasu, D., Tajaddodianfar, F., Gururajan, A., 2021. Urltran: improving phishing url detection using transformers. In: MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). IEEE, pp. 197–204.

Morgan, S., 2020. Cybercrime to cost the world $10.5 trillion annually by 2025. Cybercrime magazine. bit.ly/CybercrimeMagazine.

Odeh, A., Keshta, I., Abdelfattah, E., 2021. Phiboost-anovel phishing detection model using adaptive boosting approach. Jordanian J. Comput. Inf. Technol. 7 (01).

Oest, A., Safaei, Y., Zhang, P., Wardman, B., Tyers, K., Shoshitaishvili, Y., Doupé, A., 2020. {PhishTime}: continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 379–396.

Opara, C., Wei, B., Chen, Y., 2020. Htmlphish: enabling phishing web page detection by applying deep learning techniques on html analysis. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Opara, C., Chen, Y., Wei, B., 2023. Look before you leap: detecting phishing web pages by exploiting raw url and html characteristics. Expert Syst. Appl. 236, 121183.

OpenPhish, 2022. Openphish - phishing intelligence. https://openphish.com.

Ouyang, L., Zhang, Y., 2021. Phishing web page detection with html-level graph neural network. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, pp. 952–958.

Ozcan, A., Catal, C., Donmez, E., Senturk, B., 2023. A hybrid dnn–lstm model for detecting phishing urls. Neural Comput. Appl. 35 (7), 4957–4973.

Peng, P., Xu, C., Quinn, L., Hu, H., Viswanath, B., Wang, G., 2019. What happens after you leak your password: understanding credential sharing on phishing sites. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 181–192.

PhishTank, 2022. Phishtank - join the fight against phishing. https://www.phishtank.com/.

Richardson, L., 2022. Beautiful soup library. bit.ly/BeautifulSoup4.

Ripa, S.P., Islam, F., Arifuzzaman, M., 2021. The emergence threat of phishing attack and the detection techniques using machine learning models. In: 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), pp. 1–6.

Sabir, B., Babar, M., Gaire, R., Abuadbba, A., 2022. Reliability and robustness analysis of machine learning based phishing url detectors. IEEE Trans. Dependable Secure Comput. 01, 1–18.

Sahingoz, O.K., Buber, E., Demir, O., Diri, B., 2019. Machine learning based phishing detection from urls. Expert Syst. Appl. 117, 345–357.

Sahoo, D., Liu, C., Hoi, S.C., 2017. Malicious url detection using machine learning: a survey, arXiv preprint arXiv:1701.07179.

Sánchez-Paniagua, M., Fernández, E.F., Alegre, E., Al-Nabki, W., González-Castro, V., 2022a. Phishing url detection: a real-case scenario through login urls. IEEE Access 10, 42949–42960.

Sánchez-Paniagua, M., Fidalgo, E., Alegre, E., Alaiz-Rodríguez, R., 2022b. Phishing websites detection using a novel multipurpose dataset and web technologies features. Expert Syst. Appl. 207, 118010.

Shah, R.K., Hasan, M.K., Islam, S., Khan, A., Ghazal, T.M., Khan, A.N., 2022. Detect phishing website by fuzzy multi-criteria decision making. In: 2022 1st International Conference on AI in Cybersecurity (ICAIC), pp. 1–8.

Su, Y., 2020. Research on website phishing detection based on LSTM RNN. In: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 1, pp. 284–288.

Tang, L., Mahmoud, Q.H., 2021. A deep learning-based framework for phishing website detection. IEEE Access 10, 1509–1521.

Teraguchi, N.C.R.L.Y., Mitchell, J.C., 2004. Client-Side Defense Against Web-Based Identity Theft. Computer Science Department, Stanford University. Available: http://crypto.stanford.edu/SpoofGuard/webspoof.pdf.

Valiyaveedu, N., Jamal, S., Reju, R., Murali, V., Nithin, K., 2021. Survey and analysis on ai based phishing detection techniques. In: 2021 International Conference on Communication. In: Control and Information Sciences (ICCISc), vol. 1. IEEE, pp. 1–6.

Van Dooremaal, B., Burda, P., Allodi, L., Zannone, N., 2021. Combining text and visual features to improve the identification of cloned webpages for early phishing detection. In: The 16th International Conference on Availability, Reliability and Security, pp. 1–10.

Vecliuc, D.-D., Artene, C.-G., Tibeică, M.-N., Leon, F., 2021. An experimental study of machine learning for phishing detection. In: Asian Conference on Intelligent Information and Database Systems. Springer, pp. 427–439.

Venugopal, S., Panale, S.Y., Agarwal, M., Kashyap, R., Ananthanagu, U., 2021. Detection of malicious urls through an ensemble of machine learning techniques. In: 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). IEEE, pp. 1–6.

Vishva, E.S., Aju, D., 2021. Phisher fighter: website phishing detection system based on url and term frequency-inverse document frequency values. J. Cyber Secur. Mobil. 11 (1), 83–104.

Wang, G., 2010. Verilogo: proactive phishing detection via logo recognition.

Wang, W., Zhang, F., Luo, X., Zhang, S., 2019. Pdrcnn: precise phishing detection with recurrent convolutional neural networks. Secur. Commun. Netw. 2019.

Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M., 2020. Accurate and fast url phishing detector: a convolutional neural network approach. Comput. Netw. 178, 107275.

Xiang, G., Hong, J., Rose, C.P., Cranor, L., 2011. Cantina+ a feature-rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur. 14 (2), 1–28.

Xiao, X., Xiao, W., Zhang, D., Zhang, B., Hu, G., Li, Q., Xia, S., 2021. Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets. Comput. Secur. 108, 102372.

Yang, R., Zheng, K., Wu, B., Wu, C., Wang, X., 2021. Phishing website detection based on deep convolutional neural network and random forest ensemble learning. Sensors 21 (24), 8281.

Zhang, H., Yu, Z., Dai, G., Huang, G., Ding, Y., Xie, Y., Wang, Y., 2022. Understanding GNN computational graph: a coordinated computation, IO, and memory perspective. In: Marculescu, D., Chi, Y., Wu, C. (Eds.), Proceedings of Machine Learning and Systems, vol. 4, pp. 467–484.

Zhang, Y., Hong, J.I., Cranor, L.F., 2007. Cantina: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web, pp. 639–648.

Job van Geest is Data Scientist with experience in leading teams and collaborating with university department heads. He excels at simplifying complex technological challenges into easily applicable solutions. What sets him apart is his combination of communicative skills and analytical thinking. He has a track record of successful leadership and communication in various team settings, including leading a 500+ members student association.

His recent thesis work involved developing a novel deep learning framework for automated phishing detection, which is currently submitted for publication in a Q1 scimago journal.

He thrives in collaborative environments and value open-mindesness, resiliency, and honesty in himself and others.

Giuseppe Cascavilla is a Assistant Professor researcher at the Eindhoven University of Technology, Jheronimus Academy of Data Science, in s'Hertogenbosch, The Netherlands. Giuseppe completed his Ph.D. in Sapienza - University of Rome in 2018 with a thesis "Privacy Issues in Online Social Networks". His research interests lie mainly in monitoring cyber criminal activities surface-deep-dark web, cyber threat intelligence, protection of cyber-physical spaces, user profiling from social media activities, and reidentifying personal emotions. Giuseppe has been an active contributor and researcher in many EU FP7 and H2020 projects, such as ANITA project focusing on evolutionary and collaborative software technology for digital and cyber crime- fighting, PRoTECT to strengthen local authorities' capabilities in Public Protection, VISOR project for smart event safety, CRYMSON for the protection of Rotterdam Harbor, Marit-D, and more.

Joris Hulstijn is a researcher with a background in information systems and artificial intelligence. My research concerns the topic of Responsible AI. In addition, he is also interested in model-based auditing, continuous control monitoring and their applications, for instance in regulatory compliance. He is an active researcher in AI and Law in general, with applications in cyber security, computational auditing and regulatory supervision. I am supervising the PhD project of Kartik Chawla, which aims to build privacy rights into service contracts, in order to give users more control over their personal data.

Recently he has joined the ICR group at University of Luxembourg, with Leon van der Torre and Réka Markovich, among many others. He is working in the EXPECTATION project (Personalized Explainable Artificial Intelligence), with colleagues Amro Najjar and Igor Tchappi. The project focuses on the personalization of Explainable AI, and on the ability to provide explanations in decentralized environments, where data and algorithms are distributed over various actors. The application domain is food and health.

Nicola Zannone is an Associate Professor and Chair of the Data Protection research group, which is part of the Security cluster, in the Department of Mathematics and Computer Science at the Eindhoven University of Technology (TU/e). He received his PhD degree in Computer Science at the University of Trento in 2007 with a Ph.D. thesis on security requirements engineering, under the supervision of Prof. Fabio Massacci and Prof. John Mylopoulos. Before joining TU/e in 2008, he held a position of young researcher at the Consorzio Interuniversitario Nazionale per l'Informatica in 2006-2007 and a post-doc position at the Department of Computer Science of the University of Toronto in 2007-2008. During his PhD, he visited Center for Secure Information Systems at George Mason University in 2005 and IBM Zurich Research Laboratory in 2006. He is a member of the steering committee of the Security next generation (CSng), a community building activity of several Dutch universities focusing on cyber security.