



Article

Explainable Artificial Intelligence System for Guiding Companies and Users in Detecting and Fixing Multimedia Web Vulnerabilities on MCS Contexts

Sergio Alloza-García ¹, Iván García-Magariño ¹  and Raquel Lacuesta Gilaberte ^{2,*} 

¹ Department of Software Engineering of Artificial Intelligence, Instituto de Tecnología del Conocimiento, Complutense University of Madrid, 28040 Madrid, Spain; salloza@ucm.es (S.A.-G.); igarciam@ucm.es (I.G.-M.)

² Department of Computer Science and Engineering of Systems, Instituto de Investigación en Ingeniería de Aragón (I3A), University of Zaragoza, 50009 Zaragoza, Spain

* Correspondence: lacuesta@unizar.es; Tel.: +34-978618262

Abstract

In the evolving landscape of Mobile Crowdsourcing (MCS), ensuring the security and privacy of both stored and transmitted multimedia content has become increasingly challenging. Factors such as human mobility, device heterogeneity, dynamic topologies, and data diversity exacerbate the complexity of addressing these concerns effectively. To tackle these challenges, this paper introduces CSXAI (Crowdsourcing eXplainable Artificial Intelligence)—a novel explainable AI system designed to proactively assess and communicate the security status of multimedia resources downloaded in MCS environments. While CSXAI integrates established attack detection techniques, its primary novelty lies in its synthesis of these methods with a user-centric XAI framework tailored for the specific challenges of MCS frameworks. CSXAI intelligently analyzes potential vulnerabilities and threat scenarios by evaluating website context, attack impact, and user-specific characteristics. The current implementation focuses on the detection and explanation of three major web vulnerability classes: Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and insecure File Upload. The proposed system not only detects digital threats in advance but also adapts its explanations to suit both technical and non-technical users, thereby enabling informed decision-making before users access potentially harmful content. Furthermore, the system offers actionable security recommendations through clear, tailored explanations, enhancing users' ability to implement protective measures across diverse devices. The results from real-world testing suggest a notable improvement in users' ability to understand and mitigate security risks in MCS environments. By combining proactive vulnerability detection with user-adaptive, explainable feedback, the CSXAI framework shows promise in empowering users to enhance their security posture effectively, even with minimal cybersecurity expertise. These findings underscore the potential of CSXAI as a reliable and accessible solution for tackling cybersecurity challenges in dynamic, multimedia-driven ecosystems. Quantitative results showed high user satisfaction and interpretability ($SUS = 79.75 \pm 6.40$; USE subscales = 5.32–5.88).



Academic Editor: Michael Sheng

Received: 11 September 2025

Revised: 3 November 2025

Accepted: 4 November 2025

Published: 17 November 2025

Citation: Alloza-García, S.; García-Magariño, I.; Gilaberte, R.L. Explainable Artificial Intelligence System for Guiding Companies and Users in Detecting and Fixing Multimedia Web Vulnerabilities on MCS Contexts. *Future Internet* **2025**, *17*, 524. <https://doi.org/10.3390/fi17110524>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: contextual intelligence; cybersecurity; XAI; IoT; multimedia security; mobile crowdsourcing; vulnerability detection

1. Introduction

Cybersecurity [1] is the practice of protecting the most important systems and confidential information against digital attacks, establishing a defense line between the data it intends to protect and unauthorized access. Additionally, cybersecurity helps prevent attacks aiming to disrupt or disable the operation of a network or a device. Currently, conducting a vulnerability analysis [2] in a company is essential for maintaining a solid cybersecurity strategy. The different gaps found could lead to damage to the brand's reputation, data violations, or breaches in web multimedia protection, potentially impacting the company's continuity. However, most of these problems can be avoided by detecting vulnerabilities in advance through a vulnerability scan. This analysis evaluates possible security vulnerabilities in an organization's infrastructure that attackers could exploit. Subsequently, a detailed report is generated to help the user identify and reinforce different security weaknesses. Crowdsourcing (CS) is a practice through which a large audience is summoned [3] to contribute ideas for finding solutions to a task. This practice allows a task, typically performed by a specialist, to be carried out through an open call by numerous individuals. CS involves obtaining work, text, graphics, images, video, animations, sound, and any other digitally processable media, information, or opinions from groups of people who submit their data through the internet, social networks, and other smartphone applications. Individuals involved in CS sometimes work as paid freelancers, while others perform small tasks voluntarily. The Mobile Crowdsourcing Network (MCN) is an emerging network paradigm that harnesses the advantages of mobile devices and Crowdsourcing with human involvement. In the security realm, users need to understand the distributed system state wherever they are and collaborate with other specialists to address distributed security problems across various service providers and types of multimedia information. Currently, the gap in the literature has been analyzed, noting the existence of various systems focused on very specific cybersecurity cases. They often fail to distinguish the variety of contexts in which attacks can occur and do not use explainable language for the user. In general, explanations generated by current XAI tools are often too technical or abstract, making them inaccessible to non-expert users or security analysts who require actionable insights. This undermines trust and hinders effective response [4,5]. Moreover, many post hoc XAI methods (e.g., SHAP, LIME) introduce computational overhead, which makes real-time, scalable deployment challenging in high-speed environments. These methods present several issues, including performance variability across different AI models and the opacity of their decision-making processes, both of which hinder comprehension by human security analysts and end-users [6,7].

While numerous security tools exist, a significant gap remains in providing context-aware, role-based, and understandable security guidance tailored for non-expert users in the MCS domain, as traditional tools typically generate reports that require expert interpretation. CSXAI aims to create a framework to automate this interpretation and translation process. Therefore, our proposal aims to cover as many cyberattacks as possible using the dynamic repository of the system, while applying contextual intelligence to each case. Additionally, users will be provided with simple explanations through explainable artificial intelligence (XAI) and solutions to attempt to fix each analyzed attack.

This article primarily focuses on the cybersecurity field within multimedia and web data, emphasizing its significance in enabling users to address one of the biggest challenges in computing today: digital threats. Furthermore, we introduce a framework for detecting vulnerabilities in web environments designed for individual users or small teams (e.g., academic developers, freelancers). For users, it allows them to understand vulnerabilities on a web page and its resources in an easily comprehensible manner adapted to their needs. For technical teams, it will facilitate analyzing their development and using the

system to find possible solutions for detected vulnerabilities. Consequently, users will be able to assess the security of a website and its multimedia elements even before accessing it through an Internet of Things (IoT) device [8]. It is important to clarify that the core contribution of this work is not the creation of new standalone detection algorithms, but rather the novel integration of existing techniques with a user-centric XAI framework to address the specific gap in multimedia security within dynamic MCS environments. The remainder of this article is structured as follows: Section II introduces the current related work and addresses the existing literature gap. Section III presents our proposal for this system, while Section IV provides a detailed analysis and commentary on the results obtained during experimentation. Section V conducts a discussion on the implementation of this system, and Section VI concludes with final remarks and discusses potential future developments of the system.

2. Related Work

Cybersecurity poses a significant challenge for companies aiming to minimize or prevent diverse security incidents and corporate data breaches. Successfully achieving this enhances their resilience against cyberattacks. To achieve this, it is crucial to emphasize the importance of enhancing awareness among the personnel involved [9], whether they are employees of a company or developers of specific web multimedia resources. These individuals operate in various environments or contexts, such as education [10] or banking [11]. Numerous articles have analyzed the state of the art in these fields, revealing the existence of various cyberattacks. Crowdsourcing (CS) has garnered attention in recent years due to its potential for obtaining labeled data economically and efficiently [12]. Specifically, Mobile Crowdsourcing (MCS) has become integral to our society, with studies attempting to incorporate learning and decision frameworks [13]. Additionally, documentation comparing different contexts [14] referring to CS exists. However, one of the main vulnerabilities in this practice today [15] is the exposure of data privacy, user location [16], and their multimedia data [17], as well as their security. Therefore, it is vital to seek effective solutions, such as those provided by this system, capable of addressing these issues. Regarding the application of XAI in cybersecurity, various fields are studied [18], including intrusion detection systems, malware detection, phishing and spam detection, botnets detection, fraud detection, zero-day vulnerabilities, digital forensics, and crypto-jacking [19–21]. Furthermore, it is essential to analyze the characteristics of users receiving the information when discussing XAI. Authors in [22] stress the importance of understanding user requirements to provide tailored explanations. For example, researchers in Human–Computer Interaction (HCI) primarily focus on creating solutions meeting the needs of end-users, necessitating a comprehensive understanding of users and their actions [23], and the use of deep learning techniques for intelligent user adaptation [24]. Moreover, the work should not only focus on providing guidelines for generating high-quality explanations; explanations should be timely in the context of cybersecurity applications and offer easily comprehensible explanations based on both knowledge and data, while also offering decision-making support [24,25]. Information regarding domain knowledge and the sources of data used to reach a decision is a key component for gaining user trust [26].

Recent surveys and systematic reviews have highlighted the growing importance of explainable artificial intelligence (XAI) in cybersecurity. For instance, Masud et al. [27] emphasize the role of XAI in enhancing resilient security in the Internet of Things, while Nascita et al. [28] provide a comprehensive survey on the application of XAI for traffic classification, prediction, and intrusion detection. Similarly, Mohale and Obagbuwa [7] focus on how XAI integration can improve transparency and interpretability in intrusion detection systems. These works converge on the idea that XAI is a powerful enabler for

trust, interpretability, and transparency in complex security systems, but they primarily concentrate on network traffic, intrusion detection, and IoT contexts.

Most research in Explainable Artificial Intelligence (XAI) has focused on post hoc techniques such as LIME or SHAP, which aim to interpret the decisions of complex and opaque models—commonly referred to as “black boxes.” These methods are particularly useful for understanding why a machine learning model, such as a neural network, classifies an image or a stream of network traffic in a certain way. In contrast, CSXAI is built upon a “glass-box” model, where decisions are not derived from a statistical black box but from a transparent and deterministic simulation process executed by agents. In this context, the challenge is not to uncover the internal logic of a model, but to explain the causal sequence of a simulated attack and its operational implications. To address this, we adopted a knowledge-driven, template-based approach that enables the generation of direct, causally accurate, and reliable explanations of system behavior. These explanations are structured through our multi-layered explainability model, which adapts the level of detail according to the user profile and operational context. This strategy prioritizes clarity and factual precision in communicating security events, surpassing the limitations of traditional post hoc methods and facilitating understanding in MCS environments, where user diversity and rapid response are critical.

Moreover, we address the particular challenges of multimedia security in Mobile Crowdsourcing (MCS) contexts, where the diversity of devices, dynamic environments, and human involvement increase exposure to vulnerabilities. Our contribution lies in extending the scope of XAI-based cybersecurity solutions toward this domain. Specifically, our CSXAI system incorporates contextual intelligence to detect vulnerabilities in multimedia resources before access, adapts explanatory levels to user characteristics, and provides actionable recommendations. This approach not only fills a literature gap by focusing on multimedia security within MCS but also advances the state of the art by combining XAI-driven interpretability with practical, user-oriented guidance for mitigating digital threats.

3. System Design

In this section, we present our proposal, the CSXAI system, which is designed to detect web vulnerabilities in MCS contexts. The system’s multi-layered explainability maps directly to three defined user roles: Developer, System Administrator, and End-User. To achieve this, our design features two primary explanation profiles: Technical (Level 1): Provides a detailed, code-centric explanation tailored for Developers. Non-Technical (Level 2): An adaptive profile that provides two distinct, role-appropriate explanations—one for System Administrators (focused on system management) and one for End-Users (focused on risk analogies and safe usage).

To address the need for structured and deep explanations, we use a multi-layered explainability model. This model structures explanations into three distinct layers, ensuring that the information is not only accessible but also contextually rich and actionable for different user profiles.

The model is based on the following three layers:

- Descriptive Layer (The “What”): This foundational layer provides a high-level, non-technical summary of the findings. Its goal is to inform the user about what vulnerability was detected in simple terms. Example: “Your website is vulnerable to an attack that could let someone inject malicious code”.
- Diagnostic Layer (The “How” and “Why”): This intermediate layer offers a more detailed, technical breakdown of the vulnerability. It explains how the attack was simulated by the agent and why the system was susceptible. This layer is crucial for technical users who need to understand the root cause. Example: “An XSS agent

successfully injected the script `<script>...</script>` into a form's text field. The vulnerability exists because the server does not properly sanitize user input before rendering it on the page."

- Prescriptive Layer (The "What's Next"): The top layer provides actionable guidance. It focuses on what to do to mitigate the risk, offering concrete solutions, code examples, or best practices. This layer empowers users to move from understanding to action. Example: "To fix this, implement input sanitization using a library like DOMPurify to neutralize malicious scripts before they are stored or displayed".

The system's core function is to translate complex security data into understandable insights. This is achieved by mapping the model layers to our defined user profiles and their corresponding explainability levels as follows: (1) Technical users (Level 1 Explainability): These users require full knowledge. For them, the system generates an explanation that includes all three layers at an advanced explanation level. Crucially, it provides the detailed Diagnostic Layer, which contains the technical root cause analysis they need to debug and implement fixes effectively. (2) Non-technical users (Level 2 Explainability): For domain experts and end-users, technical jargon can be a barrier. Therefore, the system tailors their explanations by synthesizing the Descriptive Layer (what happened) with the Prescriptive Layer (what to do next). The complex "How" and "Why" of the Diagnostic Layer are abstracted, focusing on building awareness and empowering them to take protective measures. Based on the user's profile, the system assembles these components into a single, coherent, and easy-to-understand answer adapted to the required level of explainability.

This multi-layered approach ensures that CSXAI's explainability is not superficial but rather a structured process that guides the user from awareness to resolution.

Figure 1 illustrates the architecture of the proposed system, where users initiate a security analysis from their device, prompting CSXAI to generate different accesses to a website. The architecture is composed of three primary layers: the User Interface (Client Side), the CSXAI Core Engine (Server Side), and the Agent Repository. The process begins when a user submits a URL through their device. The request is sent to the Core Engine, which first performs a context analysis of the target website. Based on this context, the Engine selects the most relevant "hacker agents" from the repository. Each selected agent is then deployed to perform its specific analysis by sending probes to the target website. The agents collect raw data from these tests and return it to the Core Engine. Finally, the Engine's XAI module synthesizes these findings, applies the multi-layered explainability model to tailor the explanation to the user profile, and sends back a clear, actionable report.

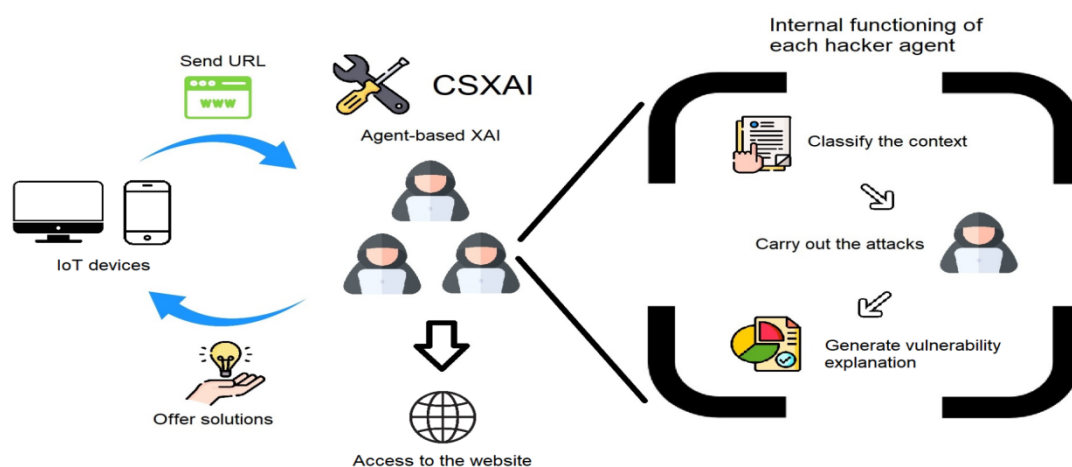


Figure 1. CSXAI system design.

Figure 2 illustrates the processes of the CSXAI system. A user initiates a verification request for a website via a mobile device using the CSXAI system. The CSXAI system conducts an analysis of the site to be evaluated, examining the available resources such as text, graphics, images, video, animations, sound, and any other digitally processable media. Subsequently, the site's context is determined, relying on both pre-established rules and user input. Following this, agent analysis begins. The required level of explainability is then determined based on the user's typology (established by the user during previous registration) and historical data gathered from prior interactions (understanding of previous explanations). Upon completion of the process, the user is prompted to confirm their understanding of the explanation provided. This response, along with the site analysis and established context, is logged for the algorithm's learning process.

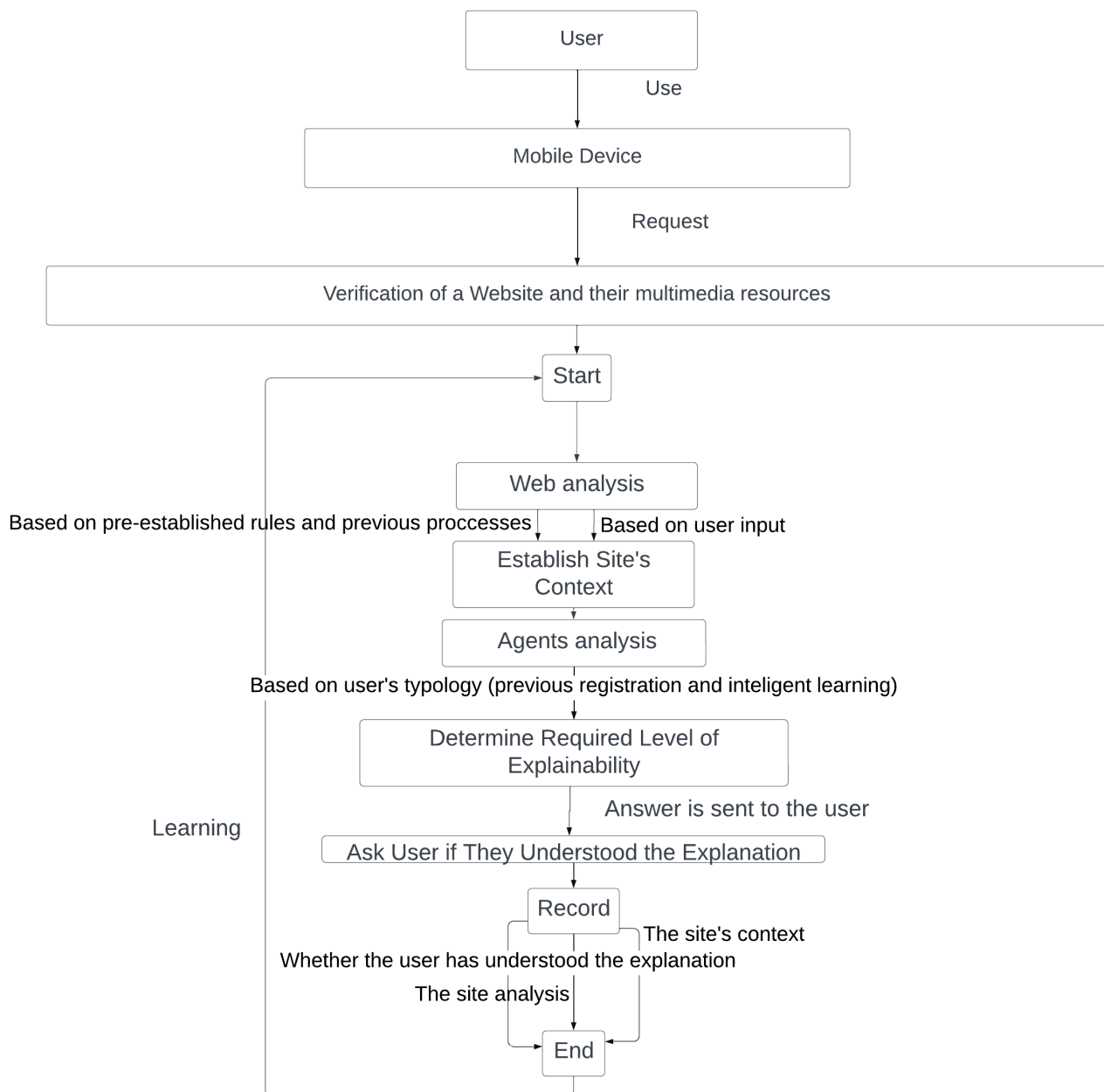


Figure 2. CSXAI system.

Figure 3 illustrates the performance of the agent analysis process. The system begins by analyzing the context and determining the need for protection, subsequently establishing

the desired security level and identifying the attacks (tests) to be executed. Following this, the system generates agents responsible for executing each selected attack. Each agent initiates the relevant requests to the website to analyze the existing real security level regarding this aspect. Once the process is completed, each agent provides feedback by explaining the conducted attack and the detected vulnerability to the intelligent system, elucidating how the resource has been compromised. Subsequently, the system compiles a comprehensive explanation of the overall process.

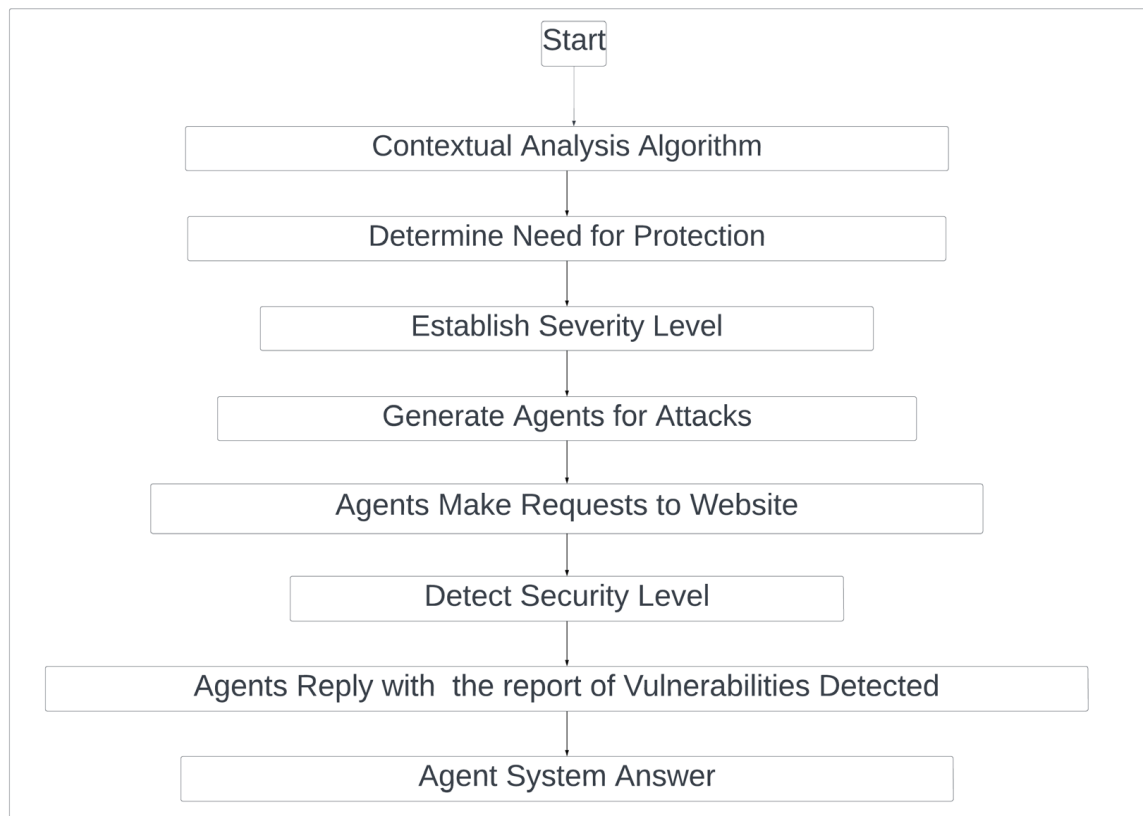


Figure 3. Agent analysis.

The system is focused on dynamically establishing a site context, determining security priorities to be analyzed (attacks and depth). The system also generates explanations of the results tailored to the specific user making the request. Furthermore, the user is provided with potential solutions to address these vulnerabilities or to protect themselves from these vulnerabilities. As justified in our analysis of the state of the art, this structured, knowledge-driven model is intentionally chosen over post hoc methods like LIME/SHAP to provide direct and causally faithful explanations of simulated security events.

The language used in the explanations is adapted to the user and should be clear and understandable. We have established three user profiles but only two levels of explainability. Technical users (first level of explainability) are comfortable with technical concepts, algorithm identification, and/or the use of algorithmic or programming languages when presenting an intelligent explanation of the problem. Secondly, domain experts possess deep knowledge and experience in specific fields relevant to the application of AI systems, such as healthcare. Finally, we have end-users who can access all types of websites but do not have specific knowledge.

When explaining security issues to non-technical users such as domain experts or end-users (second level of explainability), it is necessary to focus on improving their decision-making, providing information, and enhancing action outcomes regarding security in

accessing websites and associated resources. In these cases, emphasis should be placed on improving their security training, helping them understand, for example, the different security levels applied to different contexts, the security tests conducted in a specific context, or the necessary actions to protect themselves when accessing these resources—for example, by avoiding the use of unreliable sources. In these cases, explainability could include real examples of the consequences of accessing vulnerable resources. It is preferable for both domains to use non-technical language and avoid jargon. Overall, the key is to offer clear, concise, and contextually relevant explanations.

To contextualize the proposed framework with respect to existing solutions, it is necessary to contrast CSXAI with traditional vulnerability scanners and general-purpose XAI approaches. While conventional tools primarily focus on detection or feature attribution, they provide limited support for user-centric explanations and lack contextual adaptation. Table 1 summarizes this comparative analysis, emphasizing CSXAI's framework distinct contributions in terms of role-based explainability, context awareness, and its specific focus on multimedia and Mobile Crowdsourcing environments.

Table 1. Comparative analysis of CSXAI framework and other XAI and scanners tools.

Feature	Traditional Scanners (e.g., OWASP ZAP)	General XAI (e.g., LIME/SHAP)	CSXAI Framework
Primary Goal	Vulnerability Detection	Model Interpretation	Vulnerability Explanation and User Comprehension
Output	Technical Reports	Feature Importance Scores	Role-Based, Actionable Recommendations
Target Audience	Security Experts	Data Scientists	Technical and Non-Technical Users
Context Awareness	Limited (Scan Policies)	Model- Dependent	Core Feature (Prioritizes tests/explanations)
MCS/Multimedia Focus	General Purpose	Not Applicable	Specific Design Goal

While traditional scanners are powerful detection tools, their output is a raw, extensive list of potential vulnerabilities that requires a security expert to interpret, prioritize, and translate into actionable steps for different stakeholders (e.g., developers, administrators). The CSXAI's framework primary contribution is not to outperform these tools in raw detection, but to automate the subsequent interpretation and communication process. Our framework uses context to prioritize findings and leverages a multi-layered explainability model to deliver role-specific, actionable guidance directly to the user, bridging the gap between detection and remediation for non-expert users. To further situate our contribution, Table 1 compares CSXAI against existing classes of security frameworks, highlighting its unique focus on user-centric, context-aware explainability.

To complement the conceptual comparison, CSXAI was tested on Damn Vulnerable Web Application (DVWA) for a reduced set of 20 vulnerabilities that could be exploited with the attacks of either CSRF, File Upload, or XSS. In 90% of the evaluated cases, CSXAI properly identified the attack, provided explanations to corroborate the vulnerability, and provided hints about how to fix the errors. According to pentest-tools.com, the average precision of Burp Suite was 95% for XSS, 75% for File Inclusion, and 50% for CSRF, but they considered a bigger set of vulnerabilities from DVWA. Thus, it is arguable that this comparison is fair, since even though both experimentations were based on the

same website benchmark, the experimental setups were not the same. Apparently and arguably, our approach outperformed Burp Suite in detecting vulnerabilities considering only the ones detectable with XSS, File Upload, and CSRF attacks on average. Moreover, our approach provided reproducible attacks as explanations, while Burp Suite generally provides descriptions but does not indicate these reproducible attacks.

4. System Implementation

4.1. CSXAI Architecture

The following class diagram is associated with the system. Figure 4 shows the class diagram for the system.

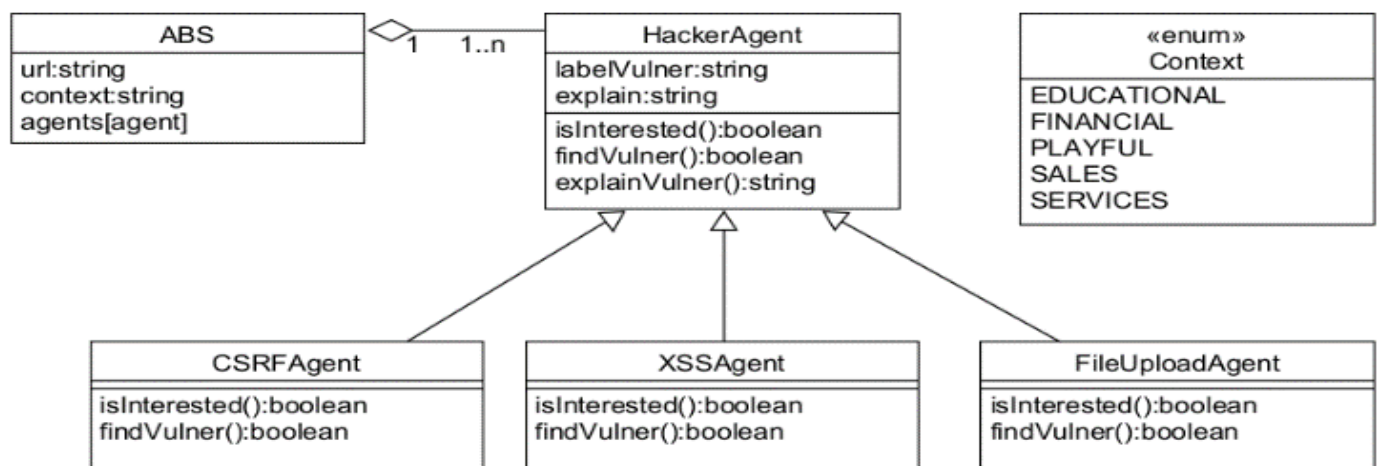


Figure 4. Class diagram of the most relevant parts of CSXAI.

As depicted in the diagram, it showcases the different agents that compose the system repository. Each one of these hacker agents corresponds to a certain type of attack (CSRF, XSS, File Upload) that a specific website can face and become vulnerable to. These agents correspond to classes of the system and are in charge of automating the different attacks that are collected in a web browser and trying to slightly violate a certain web environment.

To carry out the implementation of our proposal, we decided to carry out an agent-based XAI capable of identifying vulnerabilities in different web environments. For each of these agents that make up the system, it has been necessary to implement an interface that includes the following elements:

- **isInterested (context):** Given a context for a certain web environment, a value is returned indicating whether there is interest in hacking it. Currently, this is a Boolean value (only true or false).
- **findVulner (driver):** The agent identifies the possible vulnerabilities of the analyzed environment and stores them, adding extra information that will later be used to explain to the user. The agent receives the WebDriver with the web payload to start the operation in the web.
- **explainVulner:** The agent provides an understandable explanation for the user (in the form of a string or chain of characters) of the vulnerabilities that it has identified so far.

Agents have a web attribute, including their uniform resource locator (URL) and context, that is set in their constructor methods. During each simulation step, the different agents execute their corresponding methods as needed. However, the identification of vulnerabilities will only be carried out in cases in which the context of the web environment is interesting for the system, that is, the result returned by the element `isInterested (context)` is true. The different contexts are represented by an enum due to its exclusive nature. To

make it a more realistic scenario, each agent attacks considering a probability inferred from the context, so we may also have useful information regarding an estimation of how much time a system could take to be vulnerable. After the simulation, the explanation is composed by invoking the explain vulnerability method.

4.2. Dynamic Repository

The system offers the possibility of creating and adding new hacker agents to their current repository, with clear and simple instructions provided to carry it out. In this way, it is possible to have a large online repository available in the future so that the community can interact with these said agents. Likewise, users will be able to select sets of agents to perform various simulations and identify the different vulnerabilities found in their web environments, and add comments for each agent in the repository, as well as give them a rating and see which ones are the most trustworthy.

4.3. Selected Attacks for the Implementation

4.3.1. Background

Initially, the system allows the analysis of websites and their multimedia resources. The proposal will focus on the analysis of the following types of attacks on a specific website entered by the user, with emphasis on multimedia attacks, as follows: 1. Cross-Site Scripting (XSS) is a dangerous attack that exploits website vulnerabilities, allowing attackers to inject malicious scripts into legitimate sites. This enables them to steal personal data, session cookies, and manipulate website content. There are different types of XSS attacks, such as Reflected XSS and Stored XSS. In Reflected XSS, the payload is injected into an HTTP request parameter and displayed without validation, making it easy to modify and potentially unnoticed by users. Stored XSS involves saving input values to a storage medium and later using them in HTML documents, making all resources vulnerable to attacks. For example, if the website allows user input to interact with multimedia files (such as comments on videos or image descriptions), injection attacks like Cross-Site Scripting (XSS) could occur. Attackers might inject malicious scripts or code into these inputs, leading to the execution of unwanted actions on the user's browser, potentially compromising multimedia resources. Reflected or Stored XSS attacks can manipulate multimedia content if the website's code is vulnerable. Malicious scripts injected into the website could alter multimedia elements or redirect users to malicious sites, compromising their security. 2. DOM-based Cross-Site Scripting (DOM XSS) leverages the Document Object Model (DOM) to manipulate web page structure and content, executing malicious code on the victim's browser without their knowledge. This attack exploits the dynamic nature of DOM to modify elements and execute code on the client side. 3. File Upload attacks occur when web servers allow users to upload files without proper validation, potentially enabling the upload of malicious files that can execute code on the server. Websites that allow users to upload multimedia files can be vulnerable to various exploits if proper validation and sanitization of uploaded files are not implemented. Attackers might upload files containing malicious code or scripts that could compromise the server or other users' interactions with multimedia content. 4. Cross-Site Request Forgery (CSRF) involves tricking a user's browser into making unauthorized requests to vulnerable web applications, enabling various illegal activities such as accessing private accounts, modifying settings, or forwarding emails. CSRF attacks could be used to manipulate multimedia resources if the website's functionality related to multimedia is accessible through HTTP requests. An attacker might trick a user into unwittingly performing actions on multimedia resources, such as deleting or modifying them. 5. Insecure Direct Object References (IDORs): Inadequate access controls or improper handling of multimedia files' references could lead to IDORs vulnerabilities. Attackers

might access or manipulate multimedia resources they are not authorized to interact with, compromising their integrity or availability. 6. Weak Authentication and Authorization: If authentication and authorization mechanisms are weak or improperly implemented, attackers might gain unauthorized access to multimedia files. This could result in the theft, modification, or deletion of multimedia resources stored on the website.

Once the types of attacks that are intended to be detected through the use of the system have been described, it is convenient to address the issue of XAI, through which it will offer the user an explanation of the vulnerabilities found and their possible corrections. XAI approaches the learning of agent machines as an understandable process for users. Due to the great increase in the number of attacks against information systems, programmers are forced to adopt strategies and methodologies based on AI to protect computer software.

Within the context of computer security, explainability or understandability is essential so users are able to understand the decisions made by AI systems and interpret the results of the processes carried out. Among others, users must understand (a) the vulnerability analysis processes carried out (attacks), (b) the results, threats, or vulnerabilities existing in the system, and (c) the protection mechanisms and techniques required to applied to protect their assets.

4.3.2. Proposal

Below is a brief description of the case studies included in this article, corresponding to each of the different hacker agent attacks that make up the initial repository of the system, these being XSS, File Upload, and CSRF, explained at a higher level of detail in previous sections. All hacker agent attacks inherit elements from the HackerAgent class, through which it is possible to obtain the name of the attack, check if there is a real interest in hacking a certain website and its multimedia resources given its corresponding context, look for its vulnerabilities, and offer an understandable explanation to the user.

The main implemented agents are as follows:

- a. XSS Agent: A search is made for all the submit buttons in forms and, once located, a small script ("attackXSS.py") is introduced in the associated text fields. This script includes the insertion of a small tag ("csxai-xss-vulnerable"), used by the system to check whether or not there is an XSS vulnerability on the website being tested.
- b. File Upload Agent: A search is made for all the fields where it is possible to upload elements to the platform and, once located, the hack.php file is entered and uploaded to the page. This file includes the insertion of a small tag ("csxai-file-upload-vulnerable"), used by the system to check whether or not there is a file upload vulnerability on the website being tested, but in the case of a real attack, it could carry out malicious code execution in the web environment.
- c. CSRF Agent: A search is made for fields where the user can enter and publish information within the website, and the corresponding name attribute is obtained. With this information, a fake link is prepared and published by the user; when clicked by the current user or another user, will take the user to a certain website.

The presented agents (XSS, CSRF, and File Upload) serve as a proof of concept to showcase the capabilities of the underlying framework. These agents should be understood as illustrative examples within a modular and extensible architecture, designed to accommodate future expansion and integration of additional vulnerability detection modules across a broader landscape of web vulnerabilities.

As an example of the steps carried out by an agent, we show the code established for XSSVulner in Figure 5.

```

def findXSSVulner(url):

    """ Performs an XSS attack on the web with url parameter, and returns whether it found
    a XSS vulnerability and explanation"""

    labelVulner = 'csxai-xss-vulnerable'

    attackXSS = '<script> /* ' + labelVulner + ' */ \
        + 'document.write("Hacked by CSXAI with XSS"); </script>'

    driver = webdriver.Chrome()

    driver.get(url)

    # Perform the XSS attack to web and multimedia resources

    submitButtons = driver.find_elements_by_xpath("//input[@type = 'submit']")

    for submitButton in submitButtons:

        textInputs = driver.find_elements_by_xpath("//input[@type = 'text']")

        for textInput in textInputs:

            textInput.send_keys(attackXSS)

            time.sleep(0.5)

            submitButton.click()

    # Check if the web was compromised by us

    isVulner = labelVulner in driver.page_source

    explainFirstPart = "For each submit button of forms, I wrote in all text fields '" \
        + attackXSS + "', and I checked whether I was able to find '" + labelVulner + "'."

    if isVulner:

        explain = explainFirstPart + " I found the label in the hacked web."

    else:

        explain = explainFirstPart + " I did not found the label."

    time.sleep(0.5)

    driver.close()

    return isVulner, explain

```

Figure 5. Code for findXSSVulner Agent.

Below (Figure 6), we include the pseudo-algorithm of the most relevant excerpt for finding XSS with CSXAI approach, considering the attack generation, finding safety toggles in form of reproducible attacks, and including hints for fixing the problem.

```
def findVulner(self, driver):
    ''' Performs an XSS attack on the web with url parameter, and returns
    whether it found a XSS vulnerability and explanation'''
    labelVulner = 'csxai-xss-vulnerable'
    attackXSS='<script> document.write("Hacked by CSXAI with XSS '+labelVulner+\
    '"); </script>'
    # Perform the XSS attack
    submitButtons = driver.find_elements("xpath", "//*[@type='submit']")
    for submitButton in submitButtons:
        #textInputs = driver.find_elements_by_xpath("//*[@type='text']")
        textInputs = driver.find_elements("xpath", "//*[@type='text']")
        for textInput in textInputs:
            textInput.send_keys(attackXSS)
            time.sleep(0.2)
            submitButton.click()
    #Check if the web was vulnerated by us
    isVulner = labelVulner in driver.page_source
    explainFirstPart = "For each submit button of forms, I wrote in all text fields '" \
    +attackXSS+"', and I checked whether I was able to find '"+labelVulner+"'."
    if isVulner:
        self.explain = "The web is hackable with XSS. "+explainFirstPart+\
        " I found the label in the hacked web. \n"+\
        "TIP: Sanitize your inputs by removings tags included '<script>' tags. For example, "+\
        "in PHP, you can use 'strip_tags'."
    else:
        self.explain = "The web is robust against XSS. "+explainFirstPart+\
        " I did not found the label."
    return isVulner
```

Figure 6. CSXAI system code for findXSSVulner.

It should be remembered that, for this scenario, a secure website has been chosen to redirect users to, which may not be the case in real attacks of this type. As in the rest of the attacks, the label of the type of attack carried out (“csxai-csrf-vulnerable”) is also inserted in the damaged website.

Whether there is a vulnerability for the type of attack in question or not, the system shows an understandable message on the screen to the user, notifying the results of the test carried out and offering a possible solution to try to correct the vulnerabilities found. In this way, the initial proposal of this system (modifiable in the future) helps to establish the degree of impact of a certain attack on a concrete website within a specific context, through contextual intelligence.

4.3.3. Implementation

a. Contexts Analyzed

For this system, we decided to classify these contexts according to the definition of security and its characteristics as follows: “Computer security consists of the implementation of a set of technical measures aimed at preserving confidentiality, integrity and information availability, and can also cover other properties, such as authenticity, responsibility, reliability and non-repudiation”. Table 2 introduces some examples of categories of analysis,

depending on the specific context that have been incorporated into our first functional design.

Table 2. Contexts and their features.

	Educational	Financial	Playful	Sales
Confidentiality		×	×	
Integrity	×	×		×
Availability		×	×	×
Authenticity	×	×		
Responsibility		×		
Reliability		×	×	×
Non-repudiation		×	×	×

This classification will evolve depending on detected vulnerabilities and user demands.

b. Interface

The first version of CSXAI offers the users a very simple interface where they can enter the URL of the website they intend to analyze (Figure 7), as can be seen in the following figure. It can be used on computers or on smartphones.

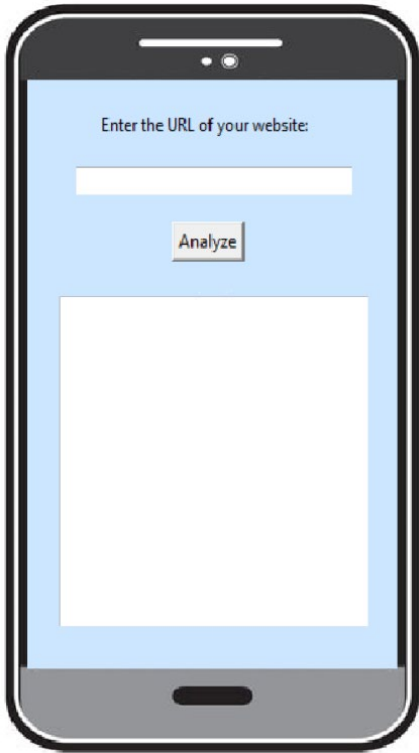


Figure 7. CSXAI mobile interface.

Once the user has entered the URL, the system will carry out a series of automated attacks (currently XSS, File Upload, and CSRF) on the website (a process that will last just a few seconds, depending on the website). Next (Figure 8), CSXAI will show the results of the vulnerability analysis carried out, as well as some advice so the user can try to correct the analyzed vulnerabilities on their website. Explanations are focused on technical users (first level of explainability).

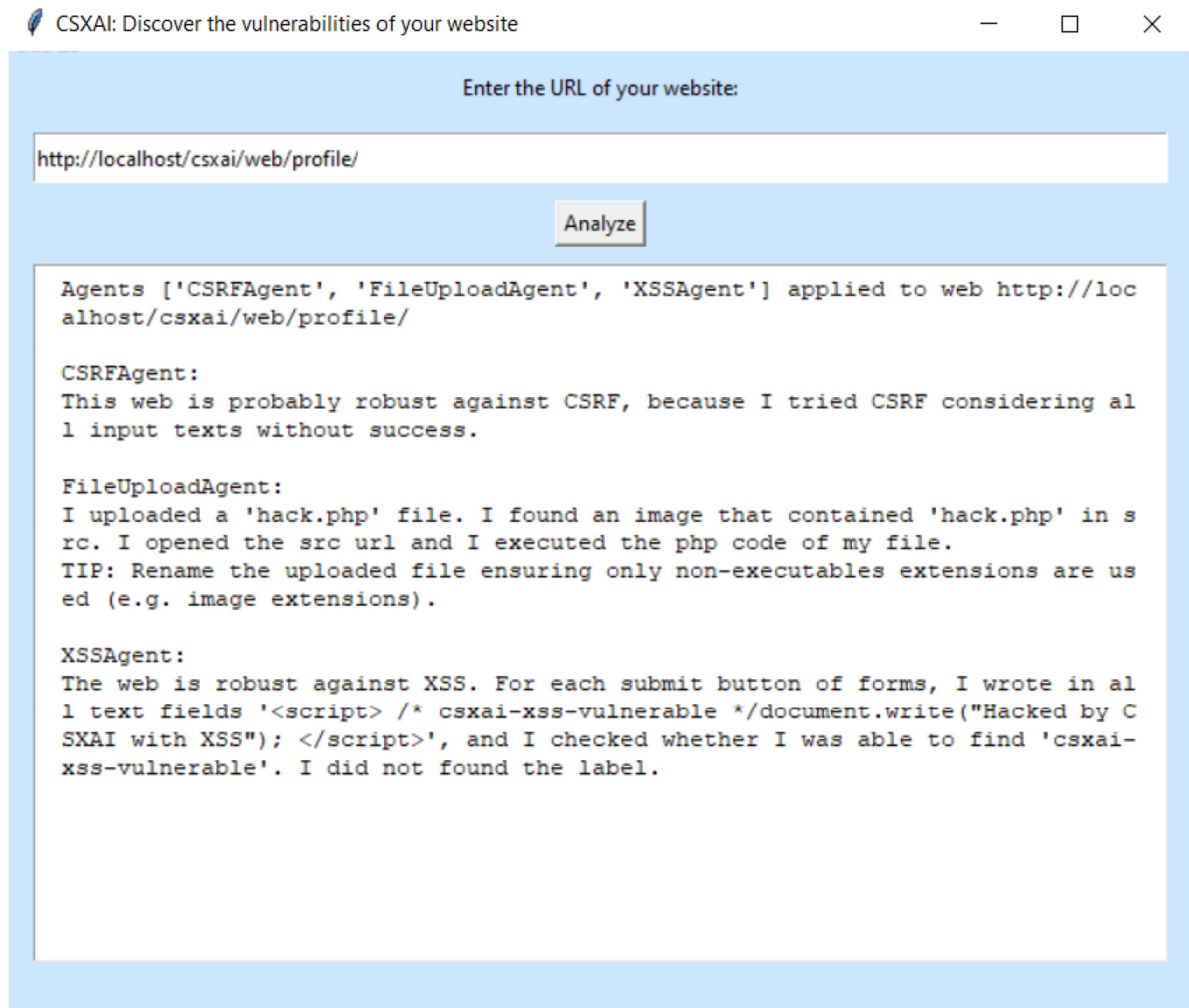


Figure 8. CSXAI results example for a “technique user profile”.

This figure presents the results of the CSXAI system about the vulnerability analysis carried out on a web-based application with the corresponding multi-agent software. More concretely, CSXAI presents the explanation of the vulnerability analysis, which contains the explanations of each agent focused on its specific attack type. Each agent has tried to perpetrate its attack, usually in different parts of the analyzed application, to check whether the corresponding attack can be easily applied to compromise the corresponding application, and describes this automatically applied experience. In this example, the first agent tried the CSRF attack. It collected the names of all the input web-based elements using “Get” parameters and created URLs that tried to exploit possible vulnerabilities through these parameters. In particular, it inserted JavaScript code that printed something inside the web-based application as proof that CSRF could be exploited through the corresponding URL. Then, it checked whether this JavaScript code had been executed when entering these URLs. In this execution example, it reported that none of these attacks was successfully achieved, as proof of some robustness against CSRF attacks.

In a similar way, the second agent tried the File Upload attack through all the File Upload fields. Using Selenium, it asked to upload a PHP file in all these fields to test

the vulnerability against File Upload attacks. More concretely, this PHP file had some instructions to execute some PHP code that printed a message that the file was being executed, to show whether the PHP file could be executed on the server. The agent checked whether there was any “src” field (commonly used in web image paths) that revealed the path of the PHP file. It actually found a src field including the PHP filename and entered the corresponding web address. It reported that the File Upload attack could be exploited as it had been able to execute the uploaded file. Then, it also provided a tip on how the developer can remove this vulnerability.

Finally, the third agent tried the XSS attack. More concretely, it wrote some JavaScript code in all the input text fields of all the forms, reporting the exact input JavaScript text used for testing this vulnerability. Then, it checked whether the JavaScript code had been executed by searching for the text that the JavaScript code tried to print. It reported robustness against XSS attacks, indicating that it did not find any evidence that any of the intended XSS attacks had been successfully applied. This output, rich in technical detail, corresponds to Level 1 Explainability and is designed for users who can understand and act upon specific technical advice.

To illustrate the system’s adaptiveness, we introduce the sentences the system shows how the same File Upload vulnerability is explained to a non-technical user (Level 2 Explainability). In that case, the responses are as follows: CSRFAgent: “This web is probably secured. The system tried to see if someone could perform actions on the site pretending to be you—like clicking buttons without your knowledge. It wasn’t possible. CAN YOU KEEP USING THE SITE? You can keep browsing without worry”. FileUploadAgent: “This web is probably NOT secured. The system was able to upload a dangerous file and make it run on the site. It’s like someone leaving a “suspicious package” inside your house and opening it. ACTION: If you manage the site “contact the person or company in charge to ensure only safe files (like images) can be uploaded, and that uploaded files cannot be executed”. ACTION: If you just use the site “avoid uploading important documents or personal information until the issue is fixed. Let the site owner know so they can review it”. CAN YOU KEEP USING THE SITE? Yes, but with caution—avoid uploading sensitive data”. XSSAgent: “This web is probably secured. The system tested whether someone could insert malicious text that changes how the page looks (like fake messages), but found no issues. CAN YOU KEEP USING THE SITE? You can keep browsing without worry”.

The explanation for the non-technical user is deliberately designed to avoid technical jargon. Rather than referencing PHP execution or server paths, it uses relatable analogies—such as leaving a suspicious package or a door unlocked—to convey the core concept of the security risk. The recommended actions are simple and role-appropriate: if the user manages the site, they are advised to contact their developer or hosting provider to restrict file execution; if the user simply interacts with the site, they are encouraged to avoid uploading sensitive content and to notify the site administrator. This clarity empowers users to take meaningful steps without needing to understand the underlying code. The system synthesizes the layers of our model, omitting the complex diagnostic details.

This dual-level presentation—tailoring explanations to both technical and non-technical profiles—is central to CSXAI’s mission of making cybersecurity understandable and actionable for all users, regardless of their background or level of expertise.

The three defined user roles (Developer, Administrator, End-User) are consistently mapped to the system’s multi-layered explainability levels (technical and non-technical).

4.4. Ethical Considerations and Safety Protocols

Recognizing the inherent risks of active vulnerability probing, the CSXAI framework is being developed and evaluated with a strong emphasis on safety and ethical principles. It

is crucial to state that all experiments and tests described in this paper have been conducted exclusively within controlled, simulated, and isolated environments.

The following protocols are enforced to ensure non-destructive testing and to lay the groundwork for responsible future application:

Benign payloads to verify without harm: The core of our safety strategy is the use of non-destructive payloads. The agents do not execute malicious attacks. Instead, they verify vulnerabilities passively. For instance, the File Upload agent uploads a harmless text file (e.g., `hack.php.txt`) containing a unique marker string. The vulnerability is confirmed if the system can later locate this marker on the server, which proves that the file filter is flawed. The goal is to demonstrate the existence of a security flaw, not to exploit it, thus ensuring no damage occurs.

Sandboxed environments: All current development and testing are performed in dedicated, isolated “sandboxes.” We use locally hosted instances of vulnerable applications like DVWA inside Docker containers. This ensures that all testing activities are completely separated from public networks and production systems, eliminating any risk of unintended impact.

Rate limiting: The framework incorporates a request throttling mechanism (rate limiting) to prevent sending an excessive number of requests in a short period. This feature, currently tested and calibrated within our controlled environments, is designed to avoid overloading a web server. In a future real-world application, this mechanism would be essential and would need to be carefully configured based on pre-agreed protocols with the owner of the target system to prevent any service disruption. Additionally, the framework is being designed to store and track previous validation attempts per target server. This means that once a vulnerability check has been successfully performed and confirmed, the system will avoid repeating the same test unnecessarily within a short time window. This optimization not only reduces server load but also improves efficiency and respects the operational integrity of the target system. The validation history will be used to determine whether a re-check is warranted, based on elapsed time, system changes, or updated configurations.

Principle of explicit consent: Although all current work is on simulated targets, the framework is being built on the principle of explicit consent. Any future use of CSXAI on live, real-world websites would be strictly conditional upon receiving prior, authorized permission from the website owners.

5. Experimentation

When carrying out the experimentation section, we test the system developed in different web environments, with the aim of verifying if it fulfills its mission, that is, whether or not it is capable of hacking them. In both cases, the system offers the user an explanation of the vulnerabilities found and their possible correction in an understandable way. The user selected for explainability has been a “technique user”. At first, some very simple test web environments were built in order to carry out the first part of the experiment and verify correct functioning. These environments propose easily hackable scenarios, and the system is capable of hacking them, including the label corresponding to the type of attacks tested on them (XSS Agent explanation). Once the correct operation has been verified, we tested it on various websites and public applications prepared to be hacked, that is, that users understand how the different attacks work and identify existing security flaws. With the intention of not conflicting with the owners or companies of several public websites, we created various personal websites (Wall, Idwall, Profile) and the recreation of a known website with the Damn Vulnerable Web App (DVWA) on a personal machine, on which the different tests for the several attacks considered were carried out. More

concretely, we used the following websites: (a) message wall website, (b) web message wall authenticated with a specific user, (c) user profile creation website, and (d) DVWA (Damn Vulnerable Web App), which includes several multimedia resources. In the specific case of DVWA, the application has been installed and our system was applied to the different interfaces that compose it (corresponding to each of the assessed attacks), having been able to exploit the XSS (Reflected) vulnerability.

5.1. Sample

The tests carried out during the experimentation of the system have been carried out with 20 users related to the world of computing, whether they are (or have been, if they have finished their studies) from higher-level programs or university degrees. This decision was made on the basis that the vast majority should have previous experience in creating web pages (even if they are simple), with the aim of them being able to understand the existing cybersecurity problems and the explanations provided by the CSXAI system. We have included four initial characterization questions in the questionnaire provided for users to verify that all the users had some experience or previous knowledge in the matter.

Thus, we obtain the following sample results: (a) of the 20 users who have participated, 17 were men and 3 were women, (b) the ages of the users were between 21 and 32 years, with an average age of approximately 25 years (24.8) and a standard deviation of 2.69, (c) of the total number of respondents, 16 were studying (or had completed) their university studies, 3 were from professional training, and 1 was completing his doctorate, and (d) all of the respondents belonged to the branch or sector of computer science, either through university or professional training.

5.2. Protocol

5.2.1. Evaluation of the Explainability of CSXAI for Detecting Vulnerabilities

The protocol carried out with the participating users was as follows: first, they were told about the idea of participating in the study and an online meeting was arranged with those interested. Next, to facilitate their experience, a real-time demonstration of the execution of the CSXAI system in the different test environments was shared with them. This demonstration was accompanied by an explanation of everything that happened during it, and possible doubts were resolved. Finally, users received and filled out the questionnaire about the explanations offered, as shown below. We asked participants to respond to questionnaires about the generated explanations. In particular, we adapted the System Usability Scale (SUS) and the Usefulness, Satisfaction, and Ease of Use (USE) scales, mainly changing ‘system’ with ‘explanations’ and ‘use’ with ‘understand’, and revising the writing to make them coherent. We also added new questions to the questionnaire, some of these questions included open-text replies so we could do a qualitative evaluation as well. The process time for each user oscillated between 15 and 20 min. SUS items were reworded to focus on the clarity, consistency, and accessibility of explanations rather than general system usability. USE items were grouped into their original subscales (Usefulness, Ease of Use, Ease of Learning, Satisfaction) and adapted to reflect the user’s experience with understanding and applying the system’s security explanations.

Two additional closed-ended and two open-ended questions were added to assess domain-specific relevance and perceived advantages in vulnerability identification and remediation.

Below (Table 3) is the total number of questions included in the questionnaire that were offered to users, including characterization questions (C1–C4) and those extra ones that we added, some of which offer text fields for the user to develop a response (E1–E2).

Table 3. Questionnaire.

Characterization Questions	
	C1. Sex. C2. Age. C3. Highest level of studies. C4. Branch or sector of said studies.
Adapted SUS Questions	
	1. I think that I would like to use this system with explanations frequently. 2. I find the explanations provided by the system unnecessarily complex. 3. I thought the explanations were easy to understand. 4. I think that I would need the support of a technical person to be able to understand these explanations. 5. I found the explanations to be well-focused. 6. I thought there was too much inconsistency in the explanations. 7. I would imagine that most people would understand these explanations very quickly. 8. I found the explanations very difficult to understand. 9. I felt very confident understanding the explanations. 10. I needed to learn a lot of things before I could understand the explanations of this system.
Adapted USE–Usefulness Questions	
	11. It helps me be more effective. 12. It helps me be more productive. 13. It is useful. 14. It gives me more control over the activities in my life. 15. It makes the things I want to accomplish easier to complete. 16. It saves my time when I use it. 17. It meets my needs. 18. It does everything I would expect it to do.
Adapted USE–Ease of Use Questions	
	19. It is easy to understand. 20. It is simple to understand. 21. It is understandable. 22. It requires the fewest steps possible to understand the vulnerabilities. 23. It is flexible. 24. Understanding it is effortless. 25. I can understand it without written instructions. 26. I do not notice any inconsistencies as I read the explanations. 27. Both occasional and regular users would like it. 28. I can recover from mistakes quickly and easily. 29. I can understand it successfully every time.

Table 3. Cont.

Adapted USE–Ease of Learning Questions	
30.	I learned to understand the explanations quickly.
31.	I easily remember how to understand the explanations.
32.	It is easy to learn to understand the explanations.
33.	I quickly became skillful at understanding the explanations.
Adapted USE–Satisfaction Questions	
34.	I am satisfied with it.
35.	I would recommend it to a friend.
36.	It is fun to use.
37.	It works the way I want it to work.
38.	It is wonderful.
39.	I feel I need to have it.
40.	It is pleasant to use.
Extra Questions	
41.	Do you find this kind of explanation useful to identify vulnerabilities in web systems?
42.	Do you think that these kinds of explanations are useful for developers to improve the security of their web systems?
Extra Open-Answer Questions	
E1.	Have you missed something in the explanations? (If so, develop your answer indicating what you missed.)
E2.	What advantages do you think there are in using these explanations to find and fix your vulnerabilities? (Develop your answer.)

5.2.2. Evaluation of the Time Execution

For this section, the time measurements corresponding to the generation of CSXAI explanations were carried out for each of the analyzed websites (Wall, IdWall, Profile, and DVWA). One hundred samples have been taken for each website, and the arithmetic mean was calculated. The hardware requirements of the machine used for the collection of time measurements are as follows: system manufacturer: This system is manufactured by Lenovo, typically produced in China. It features a 64-bit architecture and is powered by an Intel® Core™ i5-6300HQ processor running at 2.30 GHz, with a maximum speed of 2304 MHz. The CPU includes four physical cores and four logical processors.

5.2.3. Results

This section shows the results obtained during the experimentation of the system, along with commentary on the most relevant aspects. The tested environments were developed and used for the assessment of the CSXAI system, including known security vulnerabilities. In the tests performed, CSXAI agents managed to exploit these vulnerabilities. For this reason, in this section, we center analysis on aspects related to user experience.

Figure 9 shows the responses collected from users corresponding to the adaptation of the SUS questionnaire (1–10).

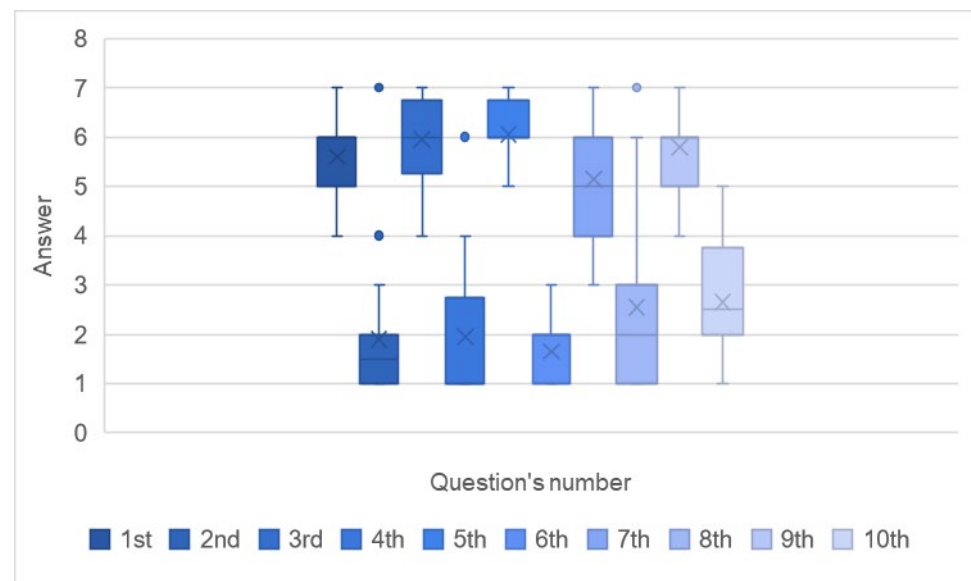


Figure 9. Adapted SUS results.

Unlike the rest of the questions in the questionnaire, the part corresponding to the adapted SUS has the peculiarity of interspersing questions that evaluate positive aspects of the system and its explanations (odd questions) with questions that evaluate the negative ones (even). The System Usability Scale (SUS) is a standardized 10-item questionnaire for assessing perceived usability. Following the standard scoring methodology, the 20 user responses were processed to yield a final score between 0 and 100 for each participant. The aggregated results for the System Usability Scale (SUS) are as follows: mean SUS score: 79.75, standard deviation (SD): 6.40, and 95% confidence interval (CI): 76.94, 82.55.

A mean score of 79.75 is significantly above the established acceptability threshold of 68, corresponding to a “Good” to “Excellent” usability rating. The standard deviation reflects a moderate spread in user scores, consistent with varied user perceptions in usability studies. Figure 10 shows the responses collected from users corresponding to the adaptation of the Usefulness section of the USE questionnaire (11–18).

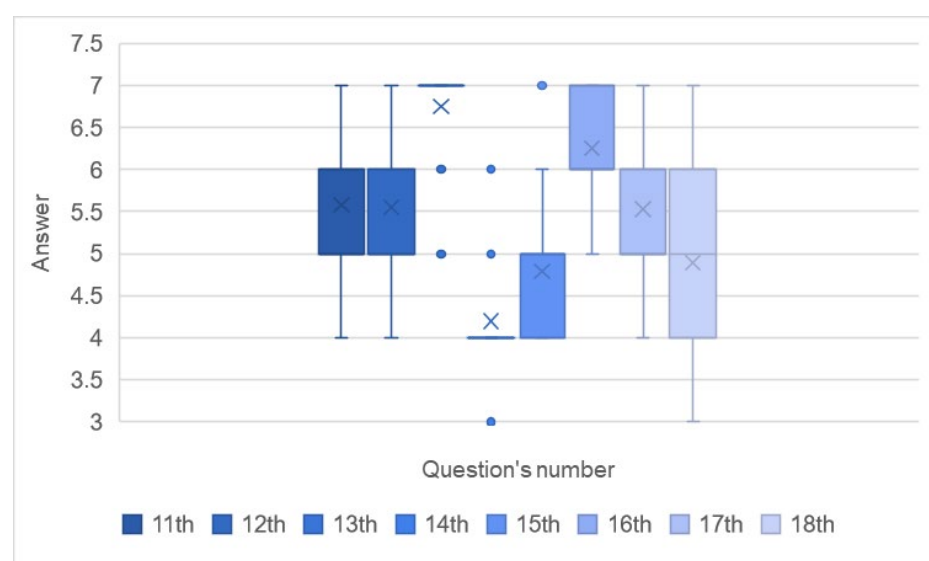


Figure 10. Adapted USE–Usefulness results.

The USE questionnaire was employed to measure four key dimensions of user experience on a 7-point Likert scale (where 1 is “Totally Disagree” and 7 is “Totally Agree”). Figure 11 shows the responses collected from users corresponding to the adaptation of the Ease of Use section of the USE questionnaire (19–29).

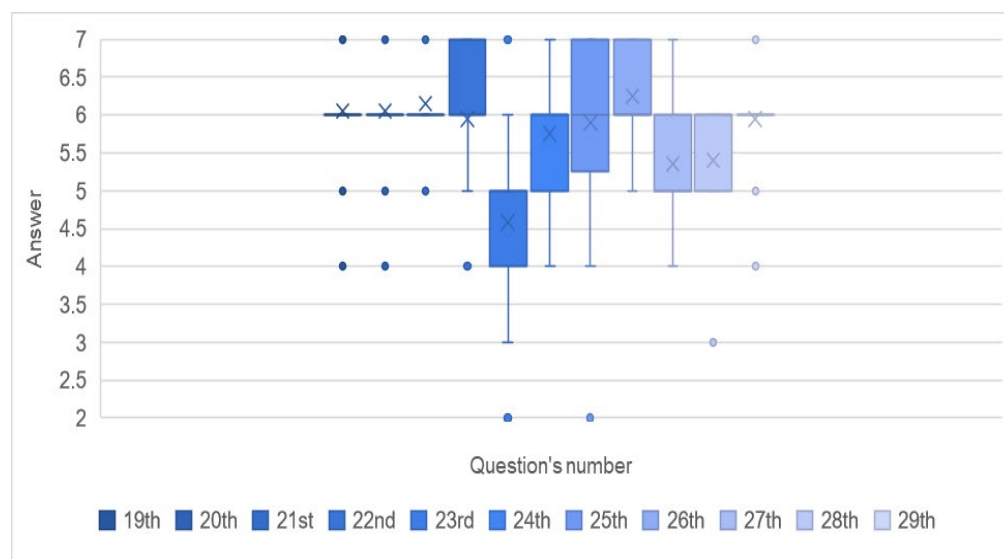


Figure 11. Adapted USE–Ease of Use results.

Figure 12 shows the responses of the users collected corresponding to the adaptation of the Ease of learning section of the USE questionnaire (30–32).

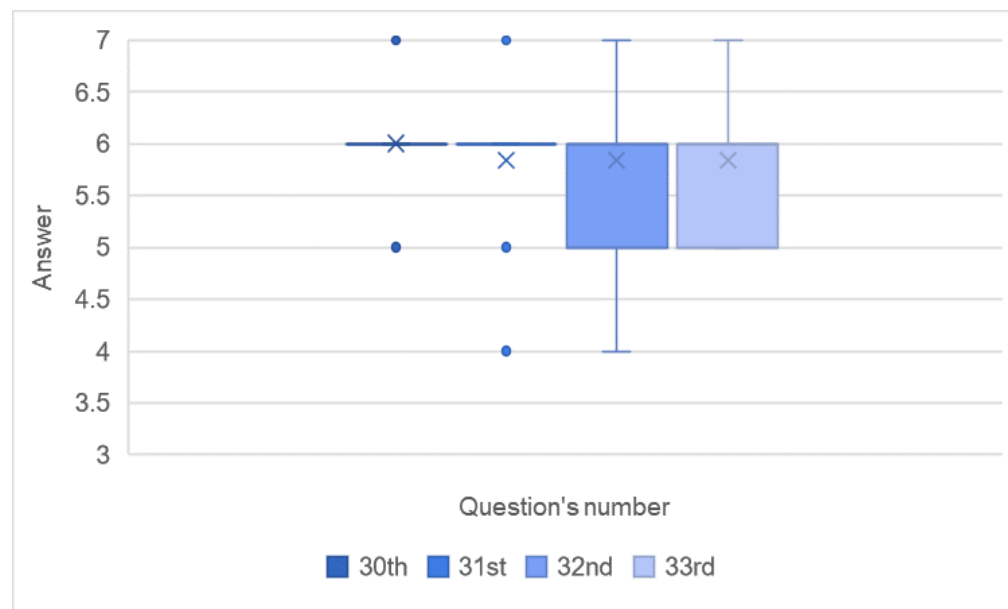


Figure 12. Adapted USE–Ease of learning results.

Figure 13 shows the responses collected from users corresponding to the adaptation of the Satisfaction section of the USE questionnaire (34–40).

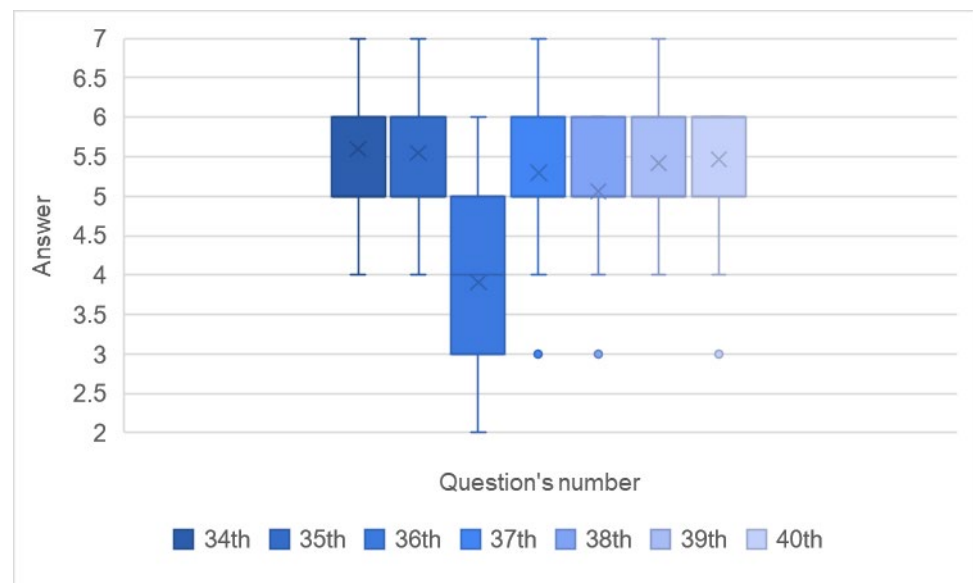


Figure 13. Adapted USE-Satisfaction results.

Figure 14 presents USE summary for each question and Table 4 presents the summary of SUS and USE (Usefulness, Satisfaction, and Ease of Use) questionnaire results.

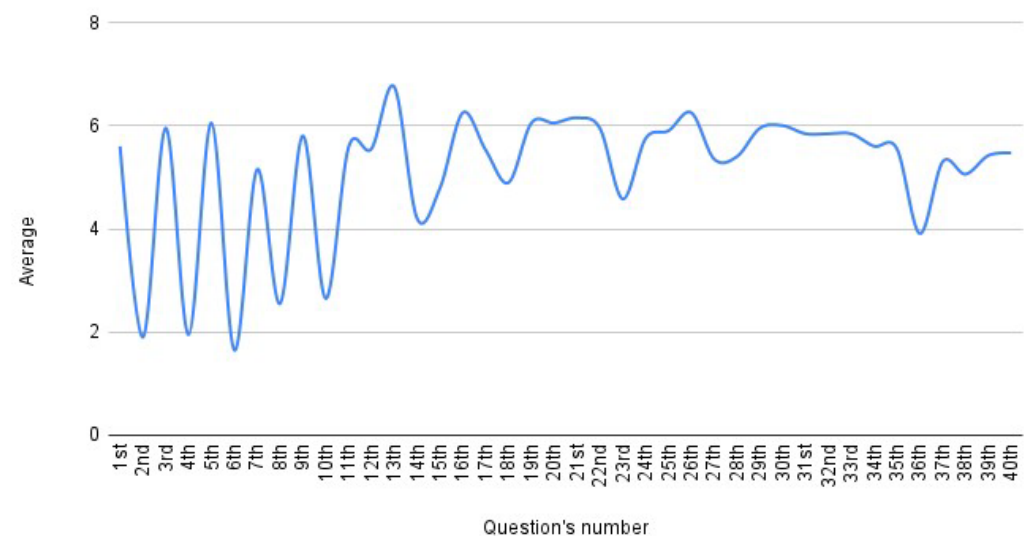


Figure 14. USE summary.

Table 4. SUS and USE results.

Subscale	Mean (Out of 7)	SD	95% CI
Usefulness (Figure 10)	5.52	0.59	[5.26, 5.77]
Ease of Use (Figure 11)	5.78	0.65	[5.49, 6.06]
Ease of Learning (Figure 12)	5.88	0.60	[5.61, 6.14]
Satisfaction (Figure 13)	5.32	0.74	[4.99, 5.64]
SUS	79.75	6.40	[76.94, 82.55]

The results from the USE questionnaire, which measures four key dimensions of user experience on a 1-to-7 scale, are summarized in Table 4. The detailed response

distributions for the subscales, Usefulness, Ease of Use, Ease of Learning, and Satisfaction, can be visualized in Figures 10, 11, 12 and 13, respectively.

The findings reveal a markedly positive user experience across all dimensions, with mean scores consistently ranging from 5.32 to 5.88. Notably, ‘Ease of Learning’ ($M = 5.88$, $SD = 0.60$) and ‘Ease of Use’ ($M = 5.78$, $SD = 0.65$) were the highest-rated dimensions, suggesting that users found the system easy to master and its interface intuitive. The standard deviations across all subscales ($SD \leq 0.74$) indicate a moderate but consistent consensus in the users’ positive perceptions. Regarding the Cronbach’s alpha justification for the SUS [29] and USE [30] questionnaires, evidence can be found in the literature. The study by Castilla et al. [31] reported a Cronbach’s alpha of 0.77 for the Spanish version of the SUS questionnaire. On the other hand, the USE questionnaire, considering the total scale, showed a Cronbach’s alpha of 0.98 according to the analysis by Gao et al. [32]. The high Cronbach’s alpha values confirm the strong reliability and internal consistency of each measurement scale, lending significant statistical validity to the results.

As a summary (Figure 13), we can conclude that users generally found that it was easy to understand the explicability from the system (3 and 9) and found the explanations well-focused (5). They also think that it is useful (13), time-saving (16), and understandable (19,20,21). Responses showed consistency (26), and the explanations were considered easy to understand (30–32). However, there are other aspects that obtained worse values, for example, “most people would understand the explanations” (7), “it gives me more control over the activities in my life” (14), “it makes the things I want to do accomplish easier to complete” (15), “it does everything I would expect it to do” (18), “it is flexible” (23), “I can recover from mistakes quickly and easily” (28), or “it is fun to use” (36). The values are not bad, but they indicate that further elaboration of more refined prototypes is needed.

Figures 15 and 16 are shown below, in which you can see the set of answers given to the first two extra questions that we have added to the questionnaire (41–42). On this scale, 1 means “Totally Disagree” and 7 means “Totally Agree”.

Do you find this kind of explanation useful to identify vulnerabilities in web systems?

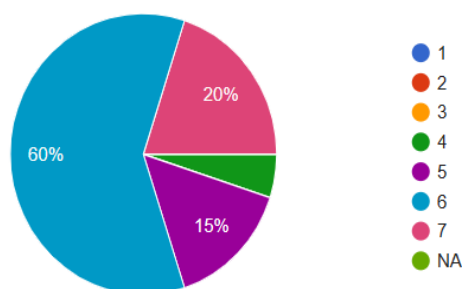


Figure 15. First extra question results.

Do you think that this kind of explanations are useful for developers to improve the security of their web systems?

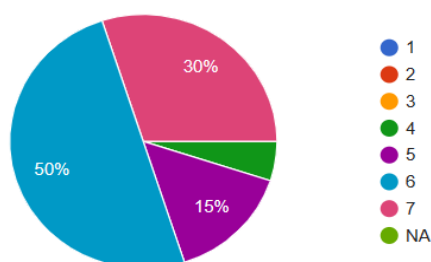


Figure 16. Second extra question results.

The results obtained from the first extra question give the system and its explanations an average of 85%, with 4 being the lowest score obtained by users. Sixty percent of users consider the explanation to be highly useful for identifying vulnerabilities.

On the other hand, the results of the second question give us an average of 86.43%, with 4 being the lowest score obtained by users. The arithmetic mean of both extra questions is 85.71%, so we can say that the test users liked the system. Fifty percent of users consider the explanation to be highly useful for developers to improve the security of their web systems.

From the analysis of questions 1–42, we can say that it is worth noting the average score of 80.4% in the questionnaire, even though certain “unpopular” results have been registered in certain questions that differ greatly from the average obtained in these specific questions. This may be due to the fact that some users have not calmly read some questions and, therefore, scored inversely. However, despite obtaining a positive assessment from users, the system that has been shown to them still has a high potential for improvement. To do this, the answers obtained from users for questions E1 and E2 of the questionnaire will be analyzed, with special emphasis on the first, which deals with what users felt was missing from the software.

E1. Have you missed something in the explanations? (If so, develop your answer indicating what you missed). Below, Table 5 collects the responses provided by users to question E1 of the questionnaire, ordered from the highest to the lowest recurrence, including possible improvements to the system.

Table 5. First open-answer extra question results.

Comment	Frequency
A short description of each of the attack types that the system offers, instead of just including its name.	8
Explanation of the possible consequences, for the web or its developer, of the different attacks in case of not correcting the vulnerabilities found.	4
A more attractive esthetic of the system, highlighting the most relevant information in another way, as well as applying better formatting.	4
Show the lines of code of the possible changes in the tip.	3
More detailed information on what the system tries to do for each of the attacks.	3
Very simple alternative explanation for the most novice users in the field of cybersecurity.	1
A short description of each of the attack types that the system offers, instead of just including its name.	8
Explanation of the possible consequences, for the web or its developer, of the different attacks in case of not correcting the vulnerabilities found.	4

As can be seen, most of the responses reflect user’s interest in knowing, by having a brief explanation of what each attack included in the system consists of, beyond how the system carries them out (which has also been requested, though to a lesser extent). On the other hand, there is a general interest in detecting the exact lines of code that web developers need to modify when applying the system’s advice, as well as improving the current version of the system (simple and somewhat rustic), so that the information is presented in a more colorful and attractive way to users. Finally, one of the users suggested

an alternative explanation using less technical vocabulary compared with the current one to encourage the use of the system among users who are less experienced in cybersecurity.

E2. What advantages do you think there are in using these explanations to find and fix your vulnerabilities? (Develop your answer). Below, Table 6 collects all the responses provided by users for question E2 of the questionnaire, which was regarding the strengths of the system, ordered from highest to lowest recurrence.

Table 6. Second open-answer extra question results.

Development of more reliable websites by increasing their security.	12
It is an immediate start-up software that guarantees significant time savings for the user who makes use of it.	6
Very useful system for users who are not so expert in cybersecurity and who are interested in this field.	3
System offers real-time results.	2
It is a very useful system for developers who do not have much knowledge about cybersecurity.	1

In general terms, users highlight the usefulness of the system to improve the security of their web environments, so that they meet at least minimum security requirements. In addition, they place great importance to the fact that carrying out this task only requires a few seconds, thereby saving time.

Regarding the execution times, Figure 17 shows the results obtained from the hundred samples for each of the four websites analyzed.

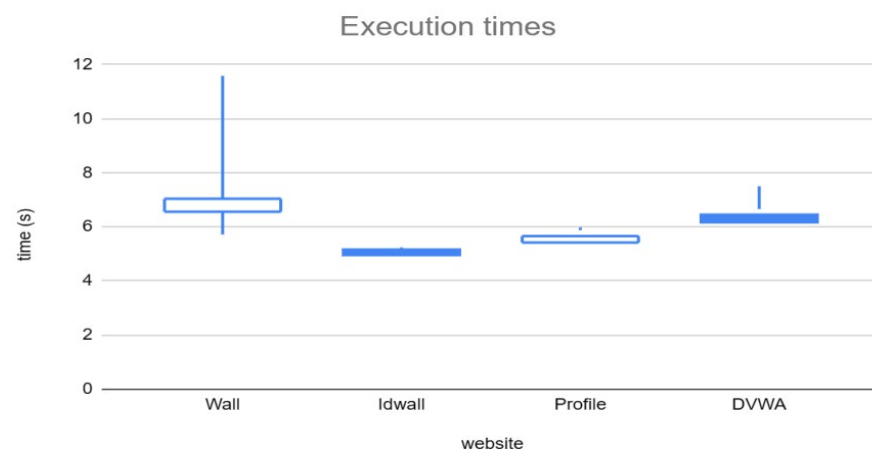


Figure 17. Execution times.

As can be seen in the graph (and taking into account the hardware requirements of the machine used for the experiments), the highest value recorded was 11.58 s (on the web Wall), corresponding to the first data collected during the tests. This contrast with respect to the rest of the values obtained is due to the fact that it was the first time that the software had been executed on the test machine and is completely eclipsed by the rest of the values between 5.5 and 7.5 s, which make an average of 6.5 s. For the rest of the websites, values obtained were certainly close to each other, between 4.6 and 6.2 s for Id Wall, between 5.1 and 7 s for Profile, and between 5.9 and 7.5 s for DVWA. Excluding the first warm-up run (11.58 s), values ranged between 4.6 s and 7.5 s, with a mean of 6.5 s, a median of 6.4 s, and an estimated variance of 0.8 s^2 ($\text{SD} \approx 0.9 \text{ s}$). The 90th and 95th percentiles were 7.3 s and 7.5 s, respectively, indicating low dispersion across trials.

6. Discussion

While prior research has established the role of Explainable Artificial Intelligence (XAI) in cybersecurity, most efforts have concentrated on domains such as intrusion detection systems, IoT security, and network traffic analysis [32]. These studies have significantly advanced interpretability and transparency in complex security infrastructures, but they generally address technical scenarios with limited direct interaction from end-users. Consequently, they often overlook the challenges of human-centered security in dynamic and multimedia-intensive contexts, such as those arising in Mobile Crowdsourcing (MCS). Our work contributes to filling this gap by proposing CSXAI (Crowdsourcing eXplainable Artificial Intelligence System), a novel system that integrates contextual intelligence with XAI to proactively detect and explain vulnerabilities in multimedia web resources within MCS environments. Unlike existing approaches that primarily focus on back-end infrastructures or specialized cybersecurity settings, CSXAI (1) extends XAI applications to the underexplored field of multimedia security in MCS, where device heterogeneity, human mobility, and data diversity create unique risks; (2) adapts explanations to user profiles, enabling both technical and non-technical users to comprehend vulnerabilities and implement mitigation strategies; (3) provides actionable guidance by not only identifying security weaknesses but also proposing concrete measures tailored to the context of the detected threat; and (4) demonstrates practical applicability through real-world testing, showing that users can effectively improve the security of their environments when supported by adaptive and comprehensible explanations.

By addressing the interplay between usability, interpretability, and security, our research advances the scientific dialog beyond current XAI-centered cybersecurity studies. It highlights the need to design systems that are not only technically robust but also accessible and empowering for diverse users, particularly in collaborative and multimedia-driven digital ecosystems.

Despite the promising results, we acknowledge several limitations in our current study that open avenues for future research. First, the experimental evaluation involved a limited cohort of 20 users, all of whom possessed computing backgrounds. Future work should include a larger and more diverse participant group to validate the system's effectiveness across broader and more heterogeneous contexts.

Second, the validation process primarily relied on subjective questionnaires aimed at assessing user comprehension and perceived utility. Although this approach aligns with the system's human-centric design objectives, we acknowledge the absence of comparative benchmarks against established automated vulnerability detection frameworks. It is important to emphasize, however, that the primary goal of this study was not to evaluate detection performance but rather to analyze the usability and interpretability of the XAI system within MCS environments. The CSXAI framework is designed to be modular, allowing for the integration of external attack detection techniques to enhance its analytical capabilities in future implementations. A crucial next step in our research will be to conduct a comparative study to benchmark CSXAI's detection capabilities and measure the efficiency of its explanatory component against traditional security reports.

To clearly define the current scope and future direction of the CSXAI framework, we provide a detailed summary of the implemented and planned vulnerability detection agents in Table 7. The system presently focuses on three major web attack classes—Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and insecure File Uploads—which are highly relevant to the dynamic nature of Mobile Crowdsourcing (MCS) platforms, where users continuously exchange and download multimedia content.

Table 7. Implemented vs. planned CSXAI agents for web vulnerability coverage in MCS contexts.

Vulnerability Category	Attack Type/Detection Target	Implemented in Current Version	Planned/In Development	Relevance to MCS Contexts
Injection and Scripting	Cross-Site Scripting (XSS)	×	—	Common in dynamic user content and comment features of MCS apps
	Cross-Site Request Forgery (CSRF)	×	—	Exploits user sessions when interacting with shared multimedia forms
	SQL/NoSQL Injection	—	×	May expose user data or multimedia metadata in back-end databases
File Handling and Uploads	Insecure File Upload	×	—	Critical for MCS platforms allowing media sharing or submissions
	MIME-Type/Content Sniffing	—	×	Prevents disguised multimedia uploads (e.g., malicious scripts)
Authentication and Access Control	Insecure Direct Object References (IDORs)	—	×	Common in media download links and shared content identifiers
	Broken Authentication/Session Fixation	—	×	Risk of hijacking mobile user sessions and tokens
	Privilege Escalation/Role Misconfiguration	—	×	Relevant to contributor vs. moderator roles in crowdsourced systems
Security Headers and Policy Configurations	Missing/Misconfigured CSP (Content Security Policy)	—	×	Limits script injection and resource loading for shared content
	Missing CORS (Cross-Origin Resource Sharing) Policy	—	×	Essential for secure API-based mobile interactions
	Insecure Cookies/Missing SameSite Flags	—	×	Impacts secure authentication on mobile browsers
Data Exposure and Privacy	Sensitive Data Exposure (e.g., EXIF, GPS in multimedia)	—	×	Directly relevant to user privacy in shared photos or videos
	Insufficient Transport Layer Security (TLS/HTTPS)	—	×	Prevents interception of multimedia uploads/downloads
Client Side and Network Attacks	Clickjacking/UI Redressing	—	×	Affects web-based MCS dashboards and previews
	Mixed Content (HTTP/HTTPS)	—	×	Common when embedding multimedia from multiple origins
Server and Application Logic	Server-Side Request Forgery (SSRF)	—	×	Exploitable when the system fetches remote multimedia resources
	Denial of Service (DoS)/Rate Limiting Abuse	—	×	Important for scalability and fair resource allocation

The next development phase will progressively expand CSXAI’s coverage to encompass a broader range of vulnerabilities affecting data handling, authentication, and multimedia sharing pipelines. These include access control flaws such as Insecure Direct Object References (IDORs), misconfigured security headers (CSP and CORS), and several other high-impact web threats (e.g., SQL/NoSQL injection, sensitive data exposure, and clickjacking). This incremental approach aims to transform CSXAI from a targeted proof of concept into a comprehensive, modular framework for explainable, user-centric web security in MCS environments.

7. Conclusions and Future Work

This paper has demonstrated the effectiveness of a vulnerability detection system tailored to Multimedia Content Sharing (MCS) environments, enabling users to swiftly and intuitively assess the security of downloaded multimedia resources, even with limited expertise in cybersecurity. This is achieved through the integration of explainable artificial intelligence (XAI), which adapts explanations both to the context of a website and to the

characteristics of each user. The CSXAI system was designed to address critical vulnerabilities and trust concerns in MCS, such as the potential injection of false data resources by attackers.

Our findings show that users generally expressed positive interest in the system and recognized its usefulness in improving their security practices. Importantly, the explanatory component was perceived as a key factor in building trust and facilitating the implementation of protective measures. However, results also indicate that there is still significant room for improvement in order to maximize the system's effectiveness. In particular, user feedback highlighted two main directions for future development: (i) expanding the explanatory content by offering alternative, simplified explanations for less experienced individuals, and (ii) enhancing the interface design to present the most relevant security information in a clearer and more visually appealing manner. These insights will guide the next iterations of CSXAI, aiming to further strengthen its role as an accessible, adaptive, and reliable framework for enhancing cybersecurity in MCS environments.

The current implementation of CSXAI focuses on detecting and explaining three major classes of web vulnerabilities—Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and insecure File Uploads. Future development will progressively extend coverage to additional attack types, including Insecure Direct Object References (IDORs) and misconfigured security headers (CSP and CORS), as well as other prevalent vulnerabilities identified in emerging MCS ecosystems. This continuous expansion aims to evolve CSXAI into a more comprehensive, adaptive, and scalable framework for proactive and explainable web security.

Our work establishes a foundation for user-centric, explainable security in MCS environments. To build upon this, we have defined a clear roadmap for future development as follows: Short-Term (Next 12 months): (1) Refine safety protocols and expand the agent repository to include detectors for common vulnerabilities like Insecure Direct Object References (IDORs) and misconfigured security headers (CSP, CORS). (2) Concurrently, run a larger, IRB-approved user study with a stratified sample and A/B testing to empirically validate the effectiveness of adaptive explanations on user comprehension and decision-making speed. Mid-Term (1–2 years): (1) Evolve the simple context model into a formal ontology that can dynamically weigh vulnerability severity based on context. (2) Launch the dynamic repository as a public web portal where the community can contribute, rate, and use new hacker agents. (3) Long-Term (3+ years): (1) Incorporate ML models to predict novel threats and automate the generation of explanations from patterns, moving beyond the current template-based system for greater adaptability and coverage. A detailed analysis of performance metrics like latency, including variance and hardware specifications, was outside the scope of this initial study but is planned for future work. Furthermore, while current execution times are practical for the tested scenarios, a detailed scalability analysis is required to assess performance on larger, more complex websites and with an expanded number of concurrent agents.

Author Contributions: Conceptualization, I.G.-M. and R.L.G.; Methodology, S.A.-G., I.G.-M., and R.L.G.; Software, S.A.-G.; Validation, I.G.-M. and R.L.G.; Investigation, S.A.-G., I.G.-M., and R.L.G.; Resources, S.A.-G. and R.L.G.; Writing—Original Draft Preparation, S.A.-G. and I.G.-M.; Writing—Review and Editing, R.L.G.; Visualization, I.G.-M.; Supervision, I.G.-M. and R.L.G.; Funding acquisition, I.G.-M. and R.L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported within the framework of the Recovery, Transformation and Resilience Plan funds, financed by the European Union (Next Generation). Strategic Project C077.23 on Cybersecurity Developed in Sparsely Populated Areas-Proyecto Estratégico de Ciberseguridad Desarrollado en Zonas Escasamente Pobladas-, implemented under the agreement between

INCIBE (Instituto Nacional de Ciberseguridad) and the University of Zaragoza. Additional support was provided by the Programa Investigo of the Madrid Autonomous Community under Grant CT36/22-08-UCM-INV, in the project “Anticipation and Defense Against Cyberattacks to Spanish Institutions Generating Explanations”. This work was also partly funded by the Spanish Ministry of Science and Innovation under Contract PID2022-136779OB-C31.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Ethics Committee CEIBA (Comité de Ética de la Investigación y Bienestar Animalof) (CEIBA2024-3470 and date of approval: 25 October 2024).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request. Due to privacy and security concerns, some of the data may not be publicly accessible to ensure the protection of sensitive information. We are committed to transparency and reproducibility of our research. Therefore, we have anonymized and curated a subset of the data, which can be accessed through a secure repository upon approval. For access to these datasets, please contact lacuesta@unizar.es.

Acknowledgments: During the preparation of this work, the author(s) used ChatGPT 4.0 and Gemini to review grammar and spelling. Some sentences were also rewritten to improve clarity, based on the authors’ original wording. The author(s) reviewed and edited the content as necessary and assume full responsibility for the final version of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MCS	Mobile Crowdsourcing
AI	Artificial Intelligence
CSXAI	Crowdsourcing eXplainable Artificial Intelligence
IoT	Internet of Things

References

1. Khan, N.F.; Yaqoob, A.; Khan, M.S.; Ikram, N. The cybersecurity behavioral research: A tertiary study. *Comput. Secur.* **2022**, *120*, 102826. [\[CrossRef\]](#)
2. Spring, J.M. An analysis of how many undiscovered vulnerabilities remain in information systems. *Comput. Secur.* **2023**, *131*, 103191. [\[CrossRef\]](#)
3. Kietzmann, J.H. Crowdsourcing: A revised definition and introduction to new research. *Bus. Horiz.* **2017**, *60*, 151–153. [\[CrossRef\]](#)
4. Larriva-Novo, X.; Sánchez-Zas, C.; Villagrà, V.; Marín-Lopez, A.; Berrocal, J. Leveraging Explainable Artificial Intelligence in Real-Time Cyberattack Identification: Intrusion Detection System Approach. *Appl. Sci.* **2023**, *13*, 8587. [\[CrossRef\]](#)
5. Rjoub, G.; Bentahar, J.; Wahab, O.; Mizouni, R.; Song, A.; Cohen, R.; Otrók, H.; Mourad, A. A Survey on Explainable Artificial Intelligence for Cybersecurity. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 5115–5140. [\[CrossRef\]](#)
6. Arreche, O.; Guntur, T.; Abdallah, M. XAI-IDS: Toward Proposing an Explainable Artificial Intelligence Framework for Enhancing Network Intrusion Detection Systems. *Appl. Sci.* **2024**, *14*, 4170. [\[CrossRef\]](#)
7. Mohale, V.; Obagbuwa, I. A systematic review on the integration of explainable artificial intelligence in intrusion detection systems to enhancing transparency and interpretability in cybersecurity. *Front. Artif. Intell.* **2025**, *8*, 1526221. [\[CrossRef\]](#)
8. Ren, Y.; Liu, W.; Liu, A.; Wang, T.; Li, A. A privacy-protected intelligent crowdsourcing application of IoT based on the reinforcement learning. *Future Gener. Comput. Syst.* **2022**, *127*, 56–69. [\[CrossRef\]](#)
9. Corallo, A.; Lazoi, M.; Lezzi, M.; Luperto, A. Cybersecurity awareness in the context of the Industrial Internet of Things: A systematic literature review. *Comput. Ind.* **2022**, *137*, 103614. [\[CrossRef\]](#)
10. Shillair, R.; Esteve-González, P.; Dutton, W.H.; Creese, S.; Nagyfejeo, E.; von Solms, B. Cybersecurity education, awareness raising, and training initiatives: National level evidence-based results, challenges, and promise. *Comput. Secur.* **2022**, *119*, 102756. [\[CrossRef\]](#)
11. Tian, S.; Zhao, B.; Olivares, R.O. Cybersecurity risks and central banks’ sentiment on central bank digital currency: Evidence from global cyberattacks. *Financ. Res. Lett.* **2022**, *53*, 103609. [\[CrossRef\]](#)

12. Shen, S.; Ji, M.; Wu, Z.; Yang, X. An optimization approach for worker selection in crowdsourcing systems. *Comput. Ind. Eng.* **2022**, *173*, 108730. [\[CrossRef\]](#)
13. Quan, J.; Wang, N. An optimized task assignment framework based on crowdsourcing knowledge graph and prediction. *Knowl.-Based Syst.* **2023**, *260*, 110096. [\[CrossRef\]](#)
14. Patterson, A.S.; Nicklin, C. L2 self-paced reading data collection across three contexts: In-person, online, and crowdsourcing. *Res. Methods Appl. Linguist.* **2023**, *2*, 100045. [\[CrossRef\]](#)
15. Wang, W.; Wang, Y.; Huang, Y.; Mu, C.; Sun, Z.; Tong, X.; Cai, Z. Privacy protection federated learning system based on blockchain and edge computing in mobile crowdsourcing. *Comput. Netw.* **2022**, *215*, 109206. [\[CrossRef\]](#)
16. Wazid, M.; Das, A.K.; Hussain, R.; Kumar, N.; Roy, S. BUAKA-CS: Blockchain-enabled user authentication and key agreement scheme for crowdsourcing system. *J. Syst. Archit.* **2022**, *123*, 102370. [\[CrossRef\]](#)
17. Li, Y.; Jeong, Y.-S.; Shin, B.-S.; Park, J.H. Crowdsensing Multimedia Data: Security and Privacy Issues. *IEEE Multimed.* **2017**, *24*, 58–66. [\[CrossRef\]](#)
18. Capuano, N.; Fenza, G.; Loia, V.; Stanzione, C. Explainable artificial intelligence in cybersecurity: A survey. *IEEE Access* **2022**, *10*, 93575–93600. [\[CrossRef\]](#)
19. Lu, X.; Tolmachev, A.; Yamamoto, T.; Takeuchi, K.; Okajima, S.; Takebayashi, T.; Maruhashi, K.; Kashima, H. Crowdsourcing Evaluation of Saliency-Based XAI Methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases; Applied Data Science Track*; Springer International Publishing: Cham, Switzerland, 2021; Volume 12979, ISBN 978-3-030-86516-0.
20. Cirqueira, D.; Nedbal, D.; Helfert, M.; Bezbradica, M. Scenario-Based Requirements Elicitation for User-Centric Explainable AI: A Case in Fraud Detection. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*; Springer International Publishing: Cham, Switzerland, 2020; pp. 321–341.
21. Garcia-Magarino, I.; Muttukrishnan, R.; Lloret, J. Human-centric AI for trustworthy IoT systems with explainable multilayer perceptrons. *IEEE Access* **2019**, *7*, 125562–125574. [\[CrossRef\]](#)
22. Lopes, P.; Silva, E.; Braga, C.; Oliveira, T.; Rosado, L. XAI Systems Evaluation: A Review of Human and Computer-Centred Methods. *Appl. Sci.* **2022**, *12*, 9423. [\[CrossRef\]](#)
23. Lv, Z.; Poesi, F.; Dong, Q.; Lloret, J.; Song, H. Deep learning for intelligent human–computer interaction. *Appl. Sci.* **2022**, *12*, 11457. [\[CrossRef\]](#)
24. Paredes, J.N.; Teze, J.C.L.; Simari, G.I.; Martinez, M.V. On the importance of domain-specific explanations in AI-based cybersecurity systems (technical report). *arXiv* **2021**, arXiv:2108.02006.
25. Martin, T. On the Need for Collaborative Intelligence in Cybersecurity. *Electronics* **2022**, *11*, 2067. [\[CrossRef\]](#)
26. Chen, S.-C. Is Artificial Intelligence New to Multimedia? *IEEE Multimed.* **2019**, *26*, 5–7. [\[CrossRef\]](#)
27. Masud, M.T.; Keshk, M.; Moustafa, N.; Linkov, I.; Emge, D.K. Explainable artificial intelligence for resilient security applications in the Internet of Things. *IEEE Open J. Commun. Soc.* **2024**, *6*, 2877–2906. [\[CrossRef\]](#)
28. Nascita, A.; Aceto, G.; Ciuonzo, D.; Montieri, A.; Persico, V.; Pescapé, A. A survey on explainable artificial intelligence for internet traffic classification and prediction, and intrusion detection. *IEEE Commun. Surv. Tutor.* **2024**, *27*, 3165–3198. [\[CrossRef\]](#)
29. Brooke, J. SUS-A quick and dirty usability scale. *Usability Eval. Ind.* **1996**, *189*, 4–7.
30. Lund, A.M. Measuring Usability with USE Questionnaire. *Usability Interface* **2001**, *8*, 3–6.
31. Castilla, D.; Jaen, I.; Suso-Ribera, C.; Garcia-Soriano, G.; Zaragoza, I.; Breton-Lopez, J.; Mira, A.; Diaz-Garcia, A.; Garcia-Palacios, A. Psychometric properties of the Spanish full and short forms of the system usability scale (SUS): Detecting the effect of negatively worded items. *Int. J. Hum.-Comput. Interact.* **2024**, *40*, 4145–4151. [\[CrossRef\]](#)
32. Gao, M.; Kortum, P.; Oswald, F. Psychometric evaluation of the USE (usefulness, satisfaction, and ease of use) questionnaire for reliability and validity. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*; Sage Publications: Los Angeles, CA, USA, 2018; Volume 62, pp. 1414–1418.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.