UDC 004.056
DOI https://doi.org/10.32782/IT/2025-1-17

Borys KUZIKOV
Candidate of Technical Sciences, Associate Professor at the Department of Computer Science, Sumy State University, 114, Kharkivska Str., Sumy, Ukraine, 40007
***ORCID:** 0000-0002-9511-5665*
***Scopus Author ID:** 55653809800*

Pavlo TYTOV
Postgraduate Student at the Department of Computer Science, Sumy State University, 114, Kharkivska Str., Sumy, Ukraine, 40007
***ORCID:** 0009-0003-6911-5463*

Oksana SHOVKOPLIAS
Candidate of Physical and Mathematical Sciences, Associate Professor at the Department of Computer Science, Sumy State University, 114, Kharkivska Str., Sumy, Ukraine, 40007
***ORCID:** 0000-0002-4596-2524*
***Scopus Author ID:** 55647364100*

Tetiana LAVRYK
Candidate of Pedagogical Sciences, Senior Lecturer at the Department of Cybersecurity, Sumy State University, Sumy State University, 114, Kharkivska Str., Sumy, Ukraine, 40007
***ORCID:** 0000-0002-7144-7059*
***Scopus Author ID:** 55674106600*

Vitalii KOVAL
Candidate of Physical and Mathematical Sciences, Senior Lecturer at the Department of Cybersecurity, Sumy State University, Sumy State University, 114, Kharkivska Str., Sumy, Ukraine, 40007
***ORCID:** 0000-0002-1593-5605*
***Scopus Author ID:** 57204958670*

Svitlana KUZIKOVA
Doctor of Psychological Sciences, Professor, Head of the Department of Psychology, Sumy State Pedagogical University named after A. S. Makarenko, 87, Romenska Str., Sumy, Ukraine, 40002
***ORCID:** 0000-0003-2574-9985*
***Scopus Author ID:** 57207304002*

To cite this article: Kuzikov, B., Tytov, P., Shovkoplias, O., Lavryk, T., Koval, V., Kuzikova, S. (2025). Vyiavlennia ta zapobihannia atakam iz vykorystanniam tekhnik maskuvannia veb-dostupnosti [Detection and prevention of accessibility cloaking attacks]. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 124–135, doi: https://doi.org/10.32782/IT/2025-1-17

DETECTION AND PREVENTION OF ACCESSIBILITY CLOAKING ATTACKS

Digital environments enable greater integration of people with disabilities into economic and social life, supported by legislative accessibility requirements. However, this progress creates new cybersecurity vulnerabilities, particularly for assistive technology users.

***Objective.** The objective of our study was to identify and analyze potential attack vectors associated with the unethical use of accessibility technologies and to develop methods for their detection and prevention, with specific focus on accessibility cloaking techniques.*

***Methods.** We conducted an analysis of popular assistive browser extensions and their detection methods, implemented proof-of-concept accessibility cloaking techniques using HTML and CSS, and evaluated the effectiveness of current automated testing tools in detecting these manipulations. Based on identified vulnerabilities, we developed a CLI application using AXE-Core for automated detection of accessibility cloaking markers.*

***Results.** Our analysis revealed multiple HTML/CSS-based techniques that create different experiences for users with and without assistive technologies, enabling malicious content to be hidden from regular users.*

While these techniques violate multiple WCAG success criteria, current automated testing tools (Wave, Axe, Lighthouse) largely failed to detect such manipulations. Our proof-of-concept detection tool, based on an agent architecture approach, successfully identified these accessibility cloaking techniques.

Conclusion. *Ensuring web resource accessibility without compromising security requires a comprehensive approach including regular security audits, additional verification of content displaying differently for different user groups, developer training, and automated detection tools. Our findings emphasize that accessibility's purpose is to make content equally accessible to all users, not to create separate or hidden experiences that can be exploited for malicious purposes.*

Key words: *accessibility, phishing, cyber security, sustainable development, digital inclusion, web equality, assistive technologies, inclusive design.*

Борис Кузіков
кандидат технічних наук, доцент кафедри комп'ютерних наук, Сумський державний університет, вул. Харківська, 144, м. Суми, Україна, 40007
ORCID: *0000-0002-9511-5665*
Scopus Author ID: *55653809800*

Павло Титов
аспірант кафедри комп'ютерних наук, Сумський державний університет, вул. Харківська, 144, м. Суми, Україна, 40007
ORCID: *0009-0003-6911-5463*

Оксана ШОВКОПЛЯС
кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук, Сумський державний університет, вул. Харківська, 144, м. Суми, Україна, 40007
ORCID: *0000-0002-4596-2524*
Scopus Author ID: *55647364100*

Тетяна Лаврик
кандидат педагогічних наук, старший викладач кафедри кібербезпеки, Сумський державний університет, вул. Харківська, 144, м. Суми, Україна, 40007
ORCID: *0000-0002-7144-7059*
Scopus Author ID: *55674106600*

Віталій Коваль
кандидат фізико-математичних наук, старший викладач кафедри кібербезпеки, Сумський державний університет, вул. Харківська, 144, м. Суми, Україна, 40007
ORCID: *0000-0002-1593-5605*
Scopus Author ID: *57204958670*

Світлана Кузікова
доктор психологічних наук, професор, завідувач кафедри психології, Сумський державний педагоічний університет імені А.С. Макаренка, Роменська, 87, м. Суми, Україна, 40002
ORCID: *0000-0003-2574-9985*
Scopus Author ID: *57207304002*

Бібліографічний опис статті: Кузіков, Б., Титов, П., Шовкопляс, О., Лавтрик, Т., Коваль, В., Кузікова С. (2025). Виявлення та запобігання атакам із використанням технік маскування веб-доступності. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 124–135, doi: https://doi.org/10.32782/IT/2025-1-17

ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ АТАКАМ ІЗ ВИКОРИСТАННЯМ ТЕХНІК МАСКУВАННЯ ВЕБ-ДОСТУПНОСТІ

Цифрові середовища забезпечують ширшу інтеграцію осіб з інвалідністю в економічне та соціальне життя, що підтримується законодавчими вимогами до доступності. Однак цей процес створює нові виклики у сфері кібербезпеки, особливо для користувачів допоміжних технологій.

Метою *нашого дослідження було ідентифікувати та проаналізувати потенційні вектори атак, пов'язані з неетичним використанням технологій доступності, а також розробити методи їх виявлення та запобігання. Основним фокусом статті є техніки, що засновані приховуванні чи спотворенні сторінки із використанням засобів підвищення доступності вебсторінок.*

Методи. *Проведено аналіз популярних браузерних розширень для забезпечення доступності та методів їх виявлення, реалізовано proof-of-concept приклади технік приховування доступності з використанням HTML та CSS, та оціннено ефективність наявних автоматизованих інструментів тестування у виявленні таких маніпуляцій. На основі виявлених вразливостей розроблено CLI-додаток з використанням AXE-Core для автоматизованого виявлення маркерів приховування доступності.*

Результати. *Аналіз виявив низку HTML/CSS-базованих технік, які створюють різне представлення контенту для користувачів з допоміжними технологіями та без них, дозволяючи приховувати шкідливий контент від звичайних користувачів. Хоча ці техніки порушують низку критеріїв успішності WCAG, поточні автоматизовані інструменти тестування (Wave, Axe, Lighthouse) здебільшого не змогли виявити такі маніпуляції. Створено інструмент виявлення, заснований на агентній архітектурі, що успішно ідентифікував ці техніки приховування доступності.*

Висновки. *Забезпечення доступності веб-ресурсів без компрометації безпеки вимагає комплексного підходу, що включає регулярні аудити безпеки, додаткову перевірку контенту, який відображається по-різному для різних груп користувачів, навчання розробників та автоматизовані інструменти виявлення. Рекомендації підкреслюють, що мета доступності – зробити контент однаково доступним для всіх користувачів, а не створювати відокремлені або приховані взаємодії, які можуть бути використані зі шкідливими намірами.*

Ключові слова: *доступність, фішинг, кібербезпека, сталий розвиток, цифрова інклюзія, рівноправність у цифровому середовищі, допоміжні технології, інклюзивний дизайн.*

1. Introduction

1.1. Motivation. Accessibility, particularly web accessibility, is mandated by many modern strategies and legislative acts. In Ukraine, the Law 'On Amendments to Certain Legislative Acts of Ukraine Regarding Ensuring Access of Persons with Special Educational Needs to Educational Services' (Law of Ukraine № 2541-VIII, 2018) is in effect; the European Union has adopted Directive (EU) 2016/2102 on the accessibility of websites and mobile applications of public sector bodies (Directive (EU) 2016/2102, 2016). Several standards exist in this field, such as Section 508 (Section 508 of the Rehabilitation Act, 1973) and EN 301 549 (ETSI EN 301 549, 2021). However, the Web Content Accessibility Guidelines (WCAG) (ISO/IEC 40500, 2012) remains the most widespread and comprehensive standard. Unfortunately, research indicates that educational resources' compliance with these standards remains relatively low (Alim, 2021; Akgül, 2020; Ismail & Kuppusamy, 2019).

The low level of educational resources' compliance with WCAG requirements is primarily due to limited awareness and lack of continuous monitoring. Currently, there are several tools, such as AXE (Deque, 2025) and WAVE (WebAIM, 2025), capable of automatically detecting up to 5–7 % of WCAG non-compliance issues (Deque, n.d.). These tools are available both commercially and free of charge, so pricing is not a limiting factor.

Website accessibility is a crucial aspect of the modern internet. Ensuring equal access to information for all users, including those with visual impairments, is key to creating an inclusive digital environment. However, improper use of accessibility technologies can lead to serious cybersecurity vulnerabilities, as malicious actors can exploit these tools to gain unauthorized system access (Jang et al., 2014; Lei et al., 2023), manipulate data, or obtain confidential information. Thus, a contradiction has emerged between insufficient awareness of accessible content creation practices, implementation of targeted web accessibility policies, and the growing number of people requiring more accessible services. This contradiction can be exploited by attackers targeting this vulnerable user group.

It is important to note that this paper focuses solely on web environment security, not mobile applications or other platforms. We concentrate on specific challenges and threats that arise in the context of web accessibility. The research examines attacks where (under insufficient control) users of specific website accessibility tools see different content than other users. The practice of providing different content to various target groups without stating the purpose of the change can be unified under the term «accessibility cloaking». This technique is hazardous because users of accessibility tools often trust specific interfaces or content customized for them, thus becoming more vulnerable to such manipulations.

1.2. **Related Works.** Using built-in accessibility features as an attack vector is one research direction. Jang (Jang et al., 2014), pioneer the research of using accessibility APIs in Windows, Android, and Mac OS operating systems as a means of security perimeter violation. Continuing this research, Lei et al. (2023) revealed a vulnerability in the password suggestion mechanism on the Android platform. Yonas Leguesse et al. (2020) demonstrated the possibility of private data disclosure. These researchers examine specific mechanisms at the operating system level. Such attacks are possible regardless of whether the user utilizes assistive technologies. Protection against these attacks is only possible at the operating system level itself. Researchers Renaud and Coles-Kemp (2022), and Wang (2017) discuss attacks using assistive tools, considering them primarily from ethical and sociocultural perspectives.

Research in this area primarily focus on the trade-off between security, usability, and accessibility. Mehralian et al. (2022), in their study address the problem of excessive accessibility in mobile applications. They propose an automated method for detecting and analyzing elements that may be too accessible, potentially creating security risks for users. This study emphasizes the importance of balancing accessibility and security in mobile application development. Goo et al., in their work 'Preserving Privacy in Assistive Technologies' (Goo et al., 2009), investigate the problem of maintaining confidentiality when using assistive technologies. They examine methods that ensure privacy for users with disabilities while providing necessary assistance. This research is crucial for understanding the balance between accessibility and personal data protection.

When addressing content substitution issues, researchers use terminology such as "content spoofing" (Jang et al., 2014), "adaptive phishing", "contextual deception" (Renaud & Coles-Kemp, 2022), "content manipulation" (Lei et al., 2023; Renaud & Coles-Kemp, 2022), "accessibility abuse", "phishing via accessibility" (Lei et al., 2023), "overlay attacks", or "content injection" (Leguesse et al., 2020). Our research subject involves displaying special content for a vulnerable category without indicating the ultimate purpose of such substitution (UAC bypass, phishing, etc.). Therefore, we propose using the term "accessibility cloaking" to describe this phenomenon more precisely.

2. Methods

The term "accessibility cloaking" describes techniques that can be used both for legitimate accessibility purposes and for potential abuse. These techniques enable the creation of different experiences for different user groups, particularly for screen reader users versus sighted users. One standard method involves using CSS to hide certain content from visual users while keeping it accessible to screen readers. Bohman and Anderson (2005) note that this approach can resolve conflicts in web development by ensuring important information remains accessible to those who need it without compromising the visual layout for other users.

However, these same techniques can be exploited by malicious actors to create targeted attacks on vulnerable user categories. It's important to note that there is usually no prior information about whether a specific user belongs to a vulnerable category. Therefore, the first step in understanding potential accessibility cloaking threats is to study methods for detecting and identifying potential targets of such attacks.

2.1. Targeting Methods. Methods for categorizing users into the target group can be divided into direct and indirect. Direct evidence includes active assistive plugins or the use of specialized browsers. Specialized browsers include text-only browsers such as WebbIE (WebbIE Web Browser, n.d.) and Lynx (Lynx Information, 2024), though they have limited usage. Users tend to prefer specialized plugins for mainstream browsers. The table 1 presents the most widespread assistive extensions for Chrome

Table 1

The most popular assistive extensions

Title	The number of installations		Augmentation of the page
	Chrome	Firefox	
Use Immersive Reader on Websites	1 000 000	-	No
High Contrast	400 000	1 007	Yes
Color Enhancer	200 000	-	Yes
OpenDyslexic	400 000	-	Yes
Speechify Text to Speech Voice Reader	1 000 000	-	Yes
Read Aloud: A Text to Speech Voice Reader	5 000 000	175 238	Yes
Vimium	500 000	41 386	No
NaturalReader – AI Text to Speech	900 000	-	Yes
Voice In	500 000	-	Yes
CrxMouse: Mouse Gestures	700 000	-	Yes
Zoom Page WE	10 000	28 111	Yes

and Firefox. The table was compiled by searching keywords in extension names and descriptions, analyzing collections of popular extensions, and and exploring related extensions suggested by browser stores. The plugins can be found by name in the official browser stores (Add-ons for Firefox, n.d.; Chrome Web Store, 2025). Plugins are ranked according to download statistics provided by the respective stores.

Plugin usage can be detected through user-agent strings and plugin APIs. However, none of the listed plugins exposed their presence through these methods. Instead, plugins made changes to pages (special classes, elements, iframes) that can uniquely identify their presence. Therefore, the most effective way to identify users from the target category is to search for specific elements and attributes on the page.

Beyond these, there are several indirect methods to detect the presence of special accessibility tools that users might utilize for website interaction. These methods are based on analyzing user behavior and specific browser or operating system settings. While none of these methods alone is sufficient for accurately detecting assistive technology usage, their combination can provide a more complete picture. First, we should highlight two methods employed in “passive” mode – on the server side, which cannot be detected by user privacy protection tools.

Special fonts. Using fonts such as OpenDyslexic, APHont, Atkinson Hyperlegible, Dyslexie, Lexie Readable, and Tiresias may indicate adaptation for users with special needs. An element can be created with font preferences from the specified list. The lowest priority is given to a server-loaded font. A request for such a font would indicate the absence of installed fonts from the control list.

Animation state detection. Users with certain cognitive disorders commonly disable animations on web pages. The detection of disabled animation states can serve as an identification marker.

In addition to “passive” methods, JavaScript tools can also be employed. The markers indicating a user belongs to the target category may include:

Using high-contrast themes or color inversion may indicate the user’s need for improved readability. This category can also include the use of other color schemes or filters.

High zoom levels (above 150 %) may indicate the user’s implementation of technologies to enhance content visibility. This category can also include text size adjustments. Significant increases in text size through browser settings may indicate visual impairments.

Intensive keyboard navigation without mouse interaction may indicate using a screen reader or other assistive technologies.

Abnormally long page dwell time may indicate using a screen reader or other assistive technologies for slow content reading. Unusual cursor movement patterns or lack of movement for extended periods may indicate the use of alternative input methods.

2.3. Cloaking Techniques. In this section, we examine technical aspects that can be used to create different experiences for users with and without assistive technologies. Understanding these methods is crucial for effectively detecting and preventing potential security threats. The section presents proof-of-concept implementations of accessibility cloaking.

2.2.1. Server-side cloaking. Server-side techniques are based on determining client characteristics primarily on the server side and delivering different content based on these characteristics. The Chameleon Attack (Elyashar et al., 2020) serves as an implementation example. Depending on user characteristics, this attack involves delivering content with varying meanings in social networks. In our case, if a user is identified as belonging to the target category, they may be served a page that differs from pages provided to other user groups. Such methods can be used for both legitimate purposes (providing more relevant content) and conducting attacks. The technique doesn’t require special markup or JavaScript usage, making it difficult to detect. The only possible approach, in this case, is co-browsing pages with agents that emulate “regular” and “assistive” clients and analyzing differences between the presented content. Of course, differences in output may be related to different data processing for different accounts, randomness in link collections, etc. Therefore, analysis requires the application of intelligent analysis methods alongside “conventional” phishing detection tools. Thus, the difference in presentation in this case is only one of the alert factors, but not decisive.

2.2.2. HTML/CSS-based Cloaking Methods. Below, we examine several approaches that allow specially formed content to be displayed for users utilizing assistive technologies. Among possible implementations, we focus on approaches that rely solely on HTML and CSS, without JavaScript usage.

A. Elements accessible only through keyboard interaction. This method is based on significant differences between the element’s content and description and hiding interaction possibilities when using a mouse (pointer).

```
<style>
   .keyboard-only { pointer-events: none;}
</style>
<a class = "keyboard-only" onclick =
"privilegedAction()" aria-label = "Hidden
actions" > & nbsp;</a>
```

B. Elements visible only to assistive tools. The following example contains a hidden input field (display: none attribute) that is not visible to most users. However, the aria-label attribute makes it accessible to screen readers. Users with visual impairments might be prompted to enter their Multi-Factor Authentication (MFA) code into this hidden field. This can result in attackers intercepting this code and gaining unauthorized access to users' accounts.

```
<style>.hidden-field { display: none; } </
style>
<form>
...
<input type=»text» name=»token» value=»1234»
aria-label=»Enter your MFA code»
class=»hidden-field» />
...
</form>
```

C. Displaying hidden blocks using aria-describedby. This method uses the aria-describedby attribute to link a hidden block with a visible element. The hidden block contains additional information that is read by screen readers but not displayed visually. This allows for providing extended descriptions or extra data for assistive technology users without altering the page's visual appearance for other users.

```
<img src = "chart.png" alt = "Financial data
for Q3" aria-describedby = "chart-desc">
<div id = "chart-desc" class = "hidden">
```

Detailed breakdown of confidential financial data, including projected earnings and market strategies. </div>

D. Moving elements outside the ViewPort. The.visually-hidden class is used to shift content beyond the visible screen area, making it invisible to users without assistive technologies. However, screen readers still read this content. Malicious actors can exploit this to direct users with visual impairments to phishing sites or other harmful resources while regular users remain unaware of such content's presence on the page.

```
<style>
   .sr-only {
      position: absolute;
      left: -9999px;
      width: 1px;
      height: 1px;
      overflow: hidden;
   }
</style>
<div class=»sr-only»>
```

This content is only available to users of screen readers. `<a href=»https://example.com/hidden-section»>Access to a hidden section</a>`
`</div>`

E. Manipulating aria-label to create alternative content. This method uses the aria-label attribute to provide alternative text descriptions for elements. Malicious actors can exploit this technique to create content that significantly differs from visually presented content. For example:

```
<a href = "https://example.com/page" aria-
label = "Access to confidential data">
Click here for more information
</a>
```

In this example, screen reader users will hear "Access to confidential data", while sighted users will see 'Click here for additional information.'

F. overlays. In this case, an invisible overlay covers part of the page. Using the opacity: 0 style, it becomes completely transparent and invisible to visual perception while remaining interactive. Users who rely on screen readers or keyboard navigation can interact with this link without realizing it redirects them to a malicious resource.

```
<style>
   .transparent-overlay {
      opacity: 0;
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
   }
</style>
<a href = "https://phishing-site.com"
class = "transparent-overlay" aria-label =
"Complete the transaction"></a>
```

G. Font size manipulation. Setting an extremely small font size (for example, 1px) to hide text from regular users while maintaining its accessibility to assistive technologies.

```
<style>
.hidden-content {
   font-size: 1px;
   color: transparent;
}
</style>
<p>The text is visible to all users. </p>
```

```
<p class = “hidden-content” > This text will
be hidden visually, but accessible to screen
readers. </p>
```

The above list is obviously incomplete, but it gives an idea of the set of patterns to pay attention to.

2.3. Countermeasures. In conclusion, it should be noted that any differences in element rendering between site versions for different users are suspicious and warrant additional verification. Performing such checks manually is time-consuming, especially considering the need to reconcile differences between two versions that might be legitimate. Automated checks face difficulties in verifying meaningful differences between versions. Therefore, the verification tool should have the following properties:

- ability to conduct regular checks and compare content between iterations;
- maximization of check automation;
- extensive use of AI tools to detect meaningful discrepancies between elements in different versions;
- to detect server-side accessibility cloaking, it is necessary to load the page at least twice, emulating “regular” and “assistive” browsers.

It is important to note that most accessibility cloaking techniques are technically valid from HTML and ARIA specifications perspective, while simultaneously violating multiple WCAG success criteria. For instance, when an aria-label vastly differs from visible text (e. g., displaying “More information” while announcing “Enter credit card details” to screen readers), it adheres to correct ARIA syntax but violates WCAG 2.5.3 Label in Name and 2.4.4 Link Purpose. Similarly, when content is moved far outside the viewport using valid CSS positioning (e.g., position: absolute; left: -9999px), it follows proper CSS implementation but contradicts WCAG 1.3.2 Meaningful Sequence and 2.4.3 Focus Order requirements. Hidden input fields using legitimate display: none properties with aria-labels may be technically correct but conflict with WCAG 1.3.1 Info and Relationships and 3.3.2 Labels or Instructions principles. These techniques create a significant challenge for automated testing tools that primarily focus on syntax validation and basic WCAG rule checking, without the capability to evaluate semantic relationships between visible and hidden content or assess the legitimacy of accessibility attribute usage.

There are numerous accessibilities checking tools, such as Wave, Axe, and Lighthouse. A test page was created to effectively test their ability to detect inconsistencies from the provided list (https://web-accessibility.sumdu.edu.ua/evil/test.html). Lighthouse 12.2.1 and Axe 4.10.2 did not detect elements contradicting accessibility standards or additional rules. Wave 3.2.7.1 detect example “G” as contrast WCAG rule violation (WCAG rule 1.4.3, insufficient contrast) and alerts to “the too-small text” (WCAG rule not provided). In addition, example “D” was alerted with “Suspicious link text” (referred to WCAG 2.4.4 Link Purpose (In Context)), but it is only keyword-based detection. Thus, standard tools show limitations in detecting these specific cases.

The inability of standard accessibility testing tools to detect such manipulations stems from several fundamental limitations. First, these tools primarily validate technical compliance with HTML and ARIA specifications – confirming syntactic correctness of aria-labels, proper implementation of hidden elements, and valid references in aria-describedby attributes. However, they cannot assess the semantic integrity of these implementations. For example, while a tool can verify that an aria-label exists, it cannot determine whether its content meaningfully matches the visible interface elements or if it potentially misleads users. Similarly, tools can confirm that hidden elements are properly concealed using valid CSS techniques but cannot evaluate whether this hiding serves a legitimate accessibility purpose or potentially obscures important content from some users.

Second, automated tools lack contextual understanding required to detect accessibility cloaking. They cannot identify mismatches between visible text and aria-labels, assess the appropriateness of hidden content placement, or evaluate the logical relationship between elements. This becomes particularly challenging in complex layouts where elements’ visual positioning might not match their programmatic order. Additionally, while tools can verify technical implementation of ARIA attributes, they cannot determine whether these implementations create a coherent and honest experience for users of assistive technologies. For instance, a technically correct aria-describedby implementation might reference content that contradicts or misrepresents the visual presentation. These limitations of automated testing highlight the need for more sophisticated verification approaches that can evaluate both technical compliance and semantic integrity of accessibility implementations.

To detect the presented examples, we developed a tool based on Axe-Core with additional verification rules (Kuzikov, n.d.). The developed tool covers only the specified inconsistencies, serving as a quick means of their detection and testing. Therefore, its overall application value is limited. Instead, the authors are developing a more comprehensive tool built on an agent-based approach. The tool's primary purpose under development is to verify educational content for Sumy State University's LMS. The input postulates include the availability of sufficient analysis tools, each of which is unimodal. Educational content is mostly multimodal. The created tool is based on agents, where each agent can run its own type of checks according to content type and presents results in a format that can be combined and harmonized with other results. The tool's sequence diagram is shown in Figure 1.

WCAG requirements were chosen for mapping results of different agents to single report. The tool is under active development; currently, two agents have been implemented: hypertext (as a wrapper for Wave and Axe), PDF (wrapper for VeraPDF), and images (contrast measurement). The rules developed within this research have been implemented as an additional agent for hypertext documents. Such a platform gives the possibility to transparently append new tools to process without code rewriting. Other example – our investigation on using a small language model to verify semantic equality aria-labels and visible content to automate checking WCAG rule 2.5.3. Checker is implemented as separate agent based on axe-code. Combining several tools into one report, as we expect, leads to raising user awareness about web accessibility standards.

3. Discussion

The growing prevalence of accessibility features in web applications is a double-edged sword. While these features are crucial for ensuring inclusiveness and equal access to online resources, malicious actors can also exploit them to attack vulnerable users. It's important to note that detecting accessibility cloaking is just one aspect of securing vulnerable users. Other potential threats, such as phishing, malware, and social engineering, must also be considered. A comprehensive approach to cybersecurity that accounts for all users' needs, including people with disabilities, is key to creating a secure and inclusive online environment.

One promising research direction is developing AI agents that can mimic the behavior of users with assistive technologies. These agents could automatically check websites for accessibility cloaking by interacting with pages in the same way screen reader users and other assistive technology users do. To detect discrepancies, the agents could analyze screen reader-parsed content and

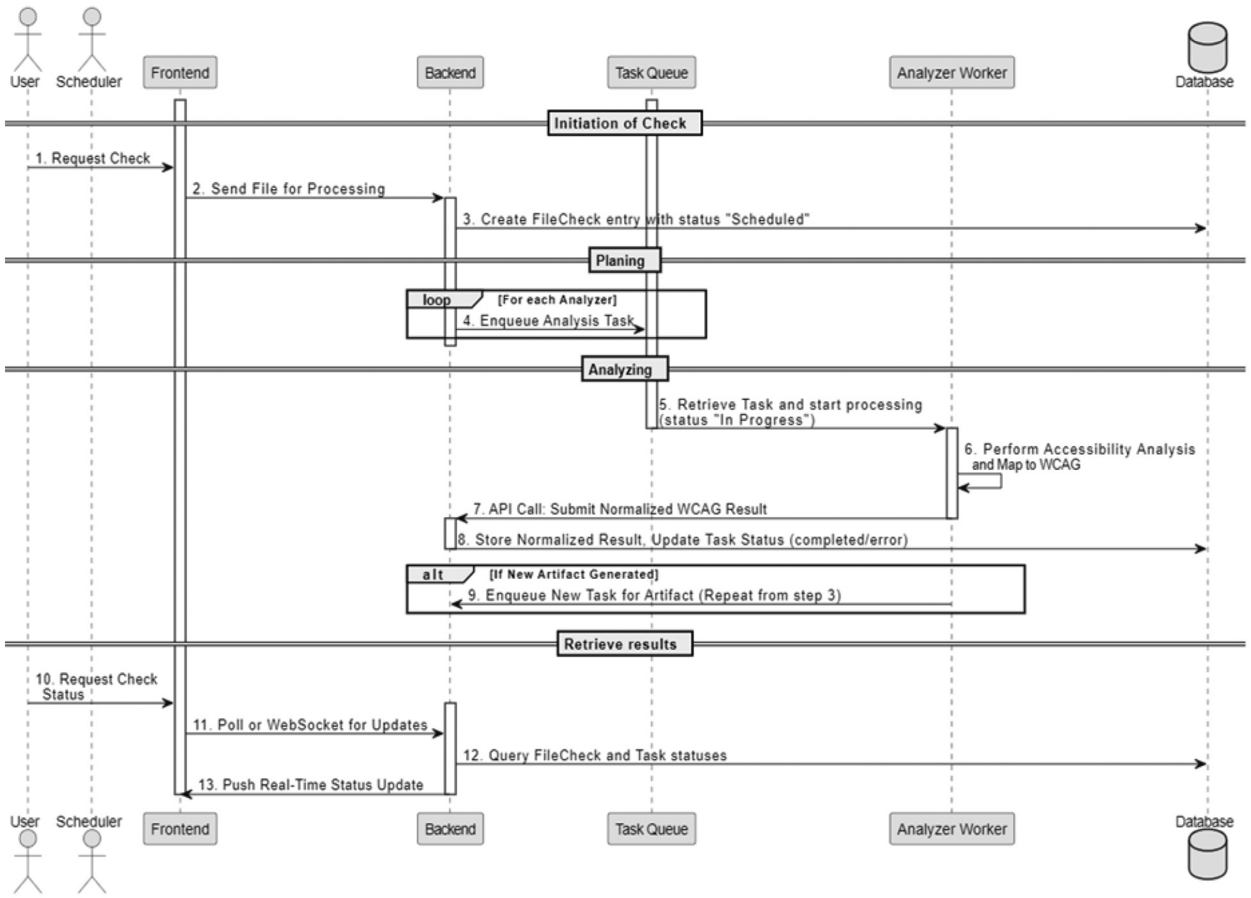
User
Scheduler
Frontend
Backend
Task Queue
Analyzer Worker
Database
Initiation of Check
1. Request Check
2. Send File for Processing
3. Create FileCheck entry with status "Scheduled"
Planing
loop [For each Analyzer]
4. Enqueue Analysis Task
Analyzing
5. Retrieve Task and start processing (status "In Progress")
6. Perform Accessibility Analysis and Map to WCAG
7. API Call: Submit Normalized WCAG Result
8. Store Normalized Result, Update Task Status (completed/error)
alt [If New Artifact Generated]
9. Enqueue New Task for Artifact (Repeat from step 3)
Retrieve results
10. Request Check Status
11. Poll or WebSocket for Updates
12. Query FileCheck and Task statuses
13. Push Real-Time Status Update

Fig. 1. Sequence Diagram for Multi-agent Accessibility Verification Tool

compare it with visually displayed content. For example, research by Sonowal and Kuppusamy (2016) showed that existing anti-phishing browser extensions have limited effectiveness for people with visual impairments. This emphasizes the need to develop specialized AI tools and algorithms that consider this user group's needs. The study proposes the MASPHID model, which helps screen reader users detect phishing sites using aural and visual similarity indicators. This approach could be valuable for integration into AI-based tools.

It's also important to note the progress in automating website accessibility. Companies like accessiBe (accessiBe, n.d.) are developing innovative solutions that use artificial intelligence to improve website accessibility automatically. Their technology can analyze web pages in real time, adjust for WCAG and ADA compliance, and adapt the interface for different types of users with disabilities. However, while such automated solutions can significantly improve many websites' accessibility, they also create new security and privacy challenges. For instance, tools that dynamically modify web page structure and content could potentially be exploited by attackers to create more sophisticated forms of accessibility cloaking. Therefore, when implementing such solutions, their impact on overall web resource security must be carefully evaluated, and additional security measures must be provided. Moreover, while automated tools can significantly ease the implementation of accessibility, they cannot completely replace manual testing and evaluation by users with disabilities.

A comprehensive approach combining automated solutions, expert evaluation, and testing by real users remains the most effective way to ensure both accessibility and security of web resources.

Summarizing the material discussed, we can formulate several recommendations for both developers and content consumers:

1. Website content should be similar for all users. High WCAG compliance is preferable to maintaining multiple separate content versions. Multiple separate pages are more challenging to keep in a consistent state. Existing problems can be further complicated if multiple content versions are maintained with divisions by other characteristics, such as language.

2. Inconsistency between visible and accessible content indirectly indicates an attack. Therefore, it should not occur, regardless of purpose.

3. Developers are responsible for content provided by their software, including user-generated content. Regular security and accessibility testing is mandatory, for example, using tools like Axe, Wave, or similar. All data must be validated and sanitized.

4. Creating a cyber-secure environment is a complex task requiring developers' and users' effort. Using specialized anti-phishing software (McAfee WebAdvisor, Avast Anti-Phishing, etc.), regular software updates, and maintaining general computer literacy and cybersecurity awareness are essential components of this process.

4. Conclusions

As a general conclusion, we emphasize that accessibility's purpose is to make content equally accessible to all users, not to create separate or hidden experiences. Any differences must be carefully considered and implemented with both accessibility and security in mind. Developers and security specialists should work together to balance accessibility and security properly. This may include:

- Regular website audits for potentially dangerous use of accessibility features.
- Implementation of additional security checks for content that displays differently for different user groups.
- Training developers in proper practices for implementing accessibility features without creating security risks.
- Using automated tools to detect potentially dangerous patterns in accessibility code.

Ensuring accessibility without compromising security is a critical responsibility for developers, requiring a balance between inclusion and protection. However, caution is necessary to avoid creating new vulnerabilities when implementing accessibility. Following best practices and regular testing will help create a secure and inclusive web space for all users. Improper or malicious use of accessibility attributes and practices in HTML can create serious vulnerabilities in web application security. Using content hidden from view but accessible only to assistive technology users, attackers can:

- Mislead users with visual impairments by directing them to phishing sites or forcing them to perform unwanted actions
- Obtain confidential information such as multi-factor authentication codes or personal data
- Execute attacks that remain undetected by most users and threat detection systems

Some solutions to the problems discussed in the paper lie in increasing developers' overall awareness of web resource accessibility. Furthermore, the mandatory implementation of automated scanners such as Wave or Axe acts as a motivating factor. The author's tool presented in the «Countermeasures» section is one of the possible instruments. The product uses an agent-based approach to combine results from multiple tools into a single harmonized report. User awareness of general principles is postulated as more critical than the number of verified rules. The analysis of Ukrainian higher education institutions' websites presented in (Kuzikov, 2024), which revealed a broad spectrum of WCAG standard non-compliance, confirms this thesis more.

4.1. Limitations and Future Work. Our study, while expanding our understanding of potential threats associated with improper use of accessibility technologies, has several limitations worth considering. First, we focused primarily on web technologies, leaving mobile applications and other platforms where similar issues may arise outside the scope. This limits the overall applicability of our findings. Second, our analysis is based on a sample of Ukrainian higher education institutions' websites (Kuzikov, 2024). While this provides valuable information, the results may not reflect the situation in other sectors, mainly e-commerce, where accessibility abuse could have more severe consequences. Finally, we focused on technical aspects, paying less attention to the social and ethical implications of using accessibility technologies in the context of cybersecurity.

Including automated verification tools as mandatory elements in the content preparation cycle for publication will increase awareness of web accessibility principles and approaches and not only help overcome the limitations of the current research but also contribute to creating a more secure and inclusive digital environment for all users, regardless of their capabilities.

Acknowledgments: The authors express their sincere gratitude to our colleague Kuzikova Svitlana, whose example continues to inspire us as she works daily to make this world more accessible. We are particularly grateful for her expert advice as a direct user of accessibility technologies, which has significantly enriched our understanding of practical implementation aspects. This work was funded by the Ministry of Education and Science of Ukraine, Grant No. 0121U109466.

BIBLIOGRAPHY:

1. Web Accessibility Solution for ADA Compliance & WCAG. *accessiBe*. : веб-сайт. URL: https://accessibe.com/ (дата звернення: 07.04.2025).

2. Add-ons for Firefox (en-GB). URL: https://addons.mozilla.org/en-GB/firefox/ (дата звернення: 07.04.2025).

3. Akgül Y. Accessibility, usability, quality performance, and readability evaluation of university websites of Turkey: a comparative study of state and private universities. *Universal Access in the Information Society*. 2021. Vol. 20, № 1. P. 157–170. DOI: https://doi.org/10.1007/S10209-020-00715-W

4. Alim S. Web Accessibility of the Top Research-Intensive Universities in the UK. *SAGE Open*>. 2021. Vol. 11, № 4. DOI: https://doi.org/10.1177/21582440211056614

5. Bohman P. R., Andersen S. A conceptual framework for accessibility tools to benefit users with cognitive disabilities. *Proc. International Cross-Disciplinary Workshop on Web Accessibility*, 2005 W4A at the World Wide Web Conference, WWW2005. 2005. P. 85–89. DOI: https://doi.org/10.1145/1061811.1061828

6. Chrome Web Store. Extensions. *Google*. URL: https://chromewebstore.google.com (дата звернення: 07.04.2025).

7. Automated Testing Identifies 57 % Digital Accessibility Issues. *Deque*. URL: https://www.deque.com/blog/automated-testing-study-identifies-57-percent-of-digital-accessibility-issues/ (дата звернення: 07.04.2025).

8. AXE: Accessibility Testing Tools and Software. *Deque*. URL: https://www.deque.com/axe/ (дата звернення: 07.04.2025).

9. Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies. URL: https://eur-lex.europa.eu/eli/dir/2016/2102/oj/eng (дата звернення: 07.04.2025).

10. Elyashar A., Uziel S., Paradise A., Puzis R. The Chameleon Attack: Manipulating Content Display in Online Social Media. *The Web Conference 2020 – Proc. World Wide Web Conference, WWW 2020*. 2020. P. 848–859. DOI: https://doi.org/10.1145/3366423.3380165

11. ETSI EN 301 549. Accessibility requirements for ICT products and services. V3.2.1, Mar. 2021.

12. Goo S. K., Irvine J. M., Andonovic I., Tomlinson A. Preserving privacy in assistive technologies. *Proc. IEEE International Conference on Communications Workshops, ICC 2009*. 2009. DOI: https://doi.org/10.1109/ICCW.2009.5208079

13. Ismail A., Kuppusamy K. S. Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites. Journal of King Saud University – *Computer and Information Sciences*. 2022. Vol. 34, № 3. P. 901–911. DOI: https://doi.org/10.1016/J.JKSUCI.2019.03.011

14. ISO/IEC 40500:2012. Information technology – W3C Web Content Accessibility Guidelines (WCAG) 2.0. Geneva, Switzerland: *International Organization for Standardization*, 2012. URL: https://www.iso.org/standard/58625.html (дата звернення: 07.04.2025).

15. Jang Y., Song C., Chung S. P., Wang T., Lee W. A11y attacks: Exploiting accessibility in operating systems. *Proc. ACM Conference on Computer and Communications Security*. 2014. P. 103–115. DOI: https://doi.org/10.1145/2660267.2660295

16. Kuzikov B. Web Accessibility HUB. URL: https://web-accessibility.sumdu.edu.ua/ (дата звернення: 07.04.2025).

17. Kuzikov B. EVIL-detector. *GitHub*. URL: https://github.com/potapuff/evil-detector (дата звернення: 07.04.2025).

18. Про внесення змін до деяких законів України щодо доступу осіб з особливими освітніми потребами до освітніх послуг : Закон України від 06.09.2018 р. № 2541-VIII. URL: https://zakon.rada.gov.ua/laws/show/2541-19 (дата звернення: 07.04.2025).

19. Leguesse Y., Vella M., Colombo C., Hernandez-Castro J. Reducing the Forensic Footprint with Android Accessibility Attacks. *Lecture Notes in Computer Science*. 2020. Vol. 12386. P. 22–38. DOI: https://doi.org/10.1007/978-3-030-59817-4_2

20. Lei, C., Ling, Z., Zhang, Y., Dong, K., Liu, K., Luo, J., & Fu, X. Do Not Give a Dog Bread Every Time He Wags His Tail: Stealing Passwords through Content Queries (CONQUER) Attacks. *Proc. Network and Distributed System Security Symposium (NDSS)*. 2023. DOI: https://doi.org/10.14722/ndss.2023.24005

21. Lynx Information. URL: https://lynx.browser.org/ (дата звернення: 27.12.2024).

22. Mehralian F., Salehnamadi N., Huq S. F., Malek S. Too Much Accessibility is Harmful! Automated Detection and Analysis of Overly Accessible Elements in Mobile Apps. *ACM International Conference Proceeding Series*. 2022. DOI: https://doi.org/10.1145/3551349.3560424

23. Renaud K., Coles-Kemp L. Accessible and Inclusive Cyber Security: A Nuanced and Complex Challenge. *SN Computer Science*. 2022. Vol. 3, № 5. P. 1–14. DOI: https://doi.org/10.1007/S42979-022-01239-1

24. IT Accessibility Laws and Policies. *Section508.gov*. URL: https://www.section508.gov/manage/laws-and-policies/ (дата звернення: 07.04.2025).

25. Sonowal G., Kuppusamy K. S. MASPHID: A Model to Assist Screen Reader Users for Detecting Phishing Sites Using Aural and Visual Similarity Measures. *ACM International Conference Proc. Series*. 2016. Vol. 25-26-August-2016. P. 87. DOI: https://doi.org/10.1145/2980258.2980443

26. Wang Y. The third wave? Inclusive privacy and security. *ACM International Conference Proc. Series*. 2017. Vol. 9. P. 122–130. DOI: https://doi.org/10.1145/3171533.3171538

27. WAVE Web Accessibility Evaluation Tools. URL: https://wave.webaim.org/ (дата звернення: 07.04.2025).

28. WebbIE Web Browser – browse the web using only text. URL: https://www.webbie.org.uk/webbrowser/index.htm (дата звернення: 07.04.2025).

REFERENCES:

1. accessiBe. (n.d.). *Web Accessibility Solution for ADA Compliance & WCAG*. Retrieved from: https://accessibe.com/

2. Add-ons for Firefox (n.d.). Retrieved from: https://addons.mozilla.org/en-GB/firefox/

3. Akgül, Y. (2020). Accessibility, usability, quality performance, and readability evaluation of university websites of Turkey: A comparative study of state and private universities. *Universal Access in the Information Society*, 20 (1). https://doi.org/10.1007/s10209-020-00715-w

4. Alim, S. (2021). Web accessibility of the top research-intensive universities in the UK. *SAGE Open*, 11(4), 215824402110566. https://doi.org/10.1177/21582440211056614

5. Bohman, P. R., & Anderson, S. (2005). A conceptual framework for accessibility tools to benefit users with cognitive disabilities. *Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A) – W4A '05*. https://doi.org/10.1145/1061811.1061828

6. Chrome Web Store, «Extensions». (2025). Google. Retrieved from: https://chromewebstore.google.com

7. Deque (n.d.) Automated Testing Identifies 57 % Digital Accessibility Issues. Retrieved from: https://www.deque.com/blog/automated-testing-study-identifies-57-percent-of-digital-accessibility-issues/

8. Deque. (2025). *Axe: Accessibility for development teams*. Retrieved from: https://www.deque.com/axe/

9. Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies. (2016, October 26). *Official Journal of the European Union*, L 327, 1–15. Retrieved from: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016L2102&amp;rid=1

10. Elyashar, A., Uziel, S., Paradise, A., & Puzis, R. (2020). The Chameleon Attack: Manipulating Content Display in Online Social Media. In *WWW ‘20: The Web Conference 2020*. ACM. https://doi.org/10.1145/3366423.3380165

11. ETSI EN 301 549 V3.2.1. (2021, March). *Accessibility requirements for ICT products and services.* European Telecommunications Standards Institute (ETSI).

12. Goo, S.K., Irvine, J.M., Andonovic, I., & Tomlinson, A. (2009). Preserving Privacy in Assistive Technologies. In *2009 IEEE International Conference on Communications Workshops*. IEEE. https://doi.org/10.1109/iccw.2009.5208079

13. Ismail, A., & Kuppusamy, K.S. (2019). Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites. *Journal of King Saud University – Computer and Information Sciences*, 34 (3). https://doi.org/10.1016/j.jksuci.2019.03.011

14. ISO/IEC 40500:2012. (2012). *Information technology – W3C Web Content Accessibility Guidelines (WCAG) 2.0*. International Organization for Standardization. Retrieved from: https://www.iso.org/standard/58625.html

15. Jang, Y., Song, C., Chung, S.P., Wang, T., & Lee, W. (2014). A11y Attacks. In *CCS’14: 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/2660267.2660295

16. Kuzikov, B. (2024). *Web Accessibility HUB*. Retrieved from: https://web-accessibility.sumdu.edu.ua/

17. Kuzikov, B. (n.d.). *EVIL-detector*. GitHub. Retrieved from: https://github.com/potapuff/evil-detector

18. Law of Ukraine № 2541-VIII “On Amendments to Certain Laws of Ukraine Regarding Access of Persons with Special Educational Needs to Educational Services”. (2018). *Vidomosti Verkhovnoi Rady (VVR), 43,* art. 345. Retrieved from: https://zakon.rada.gov.ua/laws/show/2541-19#Text

19. Leguesse, Y., Vella, M., Colombo, C., & Hernandez-Castro, J. (2020). Reducing the Forensic Footprint with Android Accessibility Attacks. In *Security and Trust Management* (pp. 22–38). Springer International Publishing. https://doi.org/10.1007/978-3-030-59817-4_2

20. Lei, C., Ling, Z., Zhang, Y., Dong, K., Liu, K., Luo, J., & Fu, X. (2023). Do Not Give a Dog Bread Every Time He Wags His Tail: Stealing Passwords through Content Queries (CONQUER) Attacks. In *Network and Distributed System Security Symposium*. Internet Society. https://doi.org/10.14722/ndss.2023.24005

21. Lynx Information (2024, December 27). Retrieved from: https://lynx.browser.org/

22. Mehralian, F., Salehnamadi, N., Huq, S.F., & Malek, S. (2022). Too Much Accessibility is Harmful! Automated Detection and Analysis of Overly Accessible Elements in Mobile Apps. In *ASE ‘22: 37th IEEE/ACM International Conference on Automated Software Engineering*. ACM. https://doi.org/10.1145/3551349.3560424

23. Renaud, K., & Coles-Kemp, L. (2022). Accessible and inclusive cyber security: A nuanced and complex challenge. *SN Computer Science*, 3 (5). https://doi.org/10.1007/s42979-022-01239-1

24. Section 508 of the Rehabilitation Act of 1973, Pub. L. No. 93-112, 87 Stat. 355 (1973). Retrieved from: https://www.section508.gov/manage/laws-and-policies

25. Sonowal, G., & Kuppusamy, K.S. (2016). MASPHID. In *ICIA-16: International Conference on Informatics and Analytics*. ACM. https://doi.org/10.1145/2980258.2980443

26. Wang, Y. (2017). The Third Wave? In NSPW ‘17: 2017 New Security Paradigms Workshop. ACM. https://doi.org/10.1145/3171533.3171538

27. WebAIM. (2025). *WAVE Web Accessibility Tool*. Wave.webaim.org. Retrieved from: https://wave.webaim.org/

28. WebbIE Web Browse. (n.d.) browse the web using only text. Retrieved from: https://www.webbie.org.uk/webbrowser/index.htm