UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Johanna Kasenurm

Web eID Authentication Window Spoofing Proof-of-Concept

Bachelor's Thesis (9 ECTS)

Supervisor:
Arnis Paršovs, PhD

Tartu 2025

Web eID Authentication Window Spoofing Proof-of-Concept

Abstract:

Phishing attacks are among the most common and effective ways to steal users' data as they manipulate users and exploit people's inattention and ignorance. The widespread use of the Web eID authentication solution makes it an important target for phishing attacks from a cybersecurity perspective. The picture-in-picture attack is a comparatively little-known attack executed in a browser environment. It represents a more sophisticated attack tactic that creates a fake pop-up window using various technical means that mimic a genuine browser or application window, but is entirely under the attacker's control. Similar tactics could be used against the Web eID authentication solution. This thesis aims to study the feasibility and effectiveness of a picture-in-picture attack against the Web eID authentication window to evaluate the security risk posed by this phishing attack. For this purpose, the bachelor's thesis aimed to create a web application demonstrating the picture-in-picture phishing attack against the Web eID authentication window, aiming to obtain the user's PIN1. A user study was conducted to test the effectiveness of the created proof-of-concept implementation. The results show that only two out of ten participants could detect the attack. This proves that a picture-in-picture attack against the Web eID authentication window can be very effective in practice.

Keywords: Web eID, picture-in-picture phishing attack, UI window

CERCS: T120 Systems engineering, computer technology

Web eID autentimisakna võltsimisrünnaku kontseptsiooni tõendus

Lühikokkuvõte:

Õngitsusrünnakud on ühed levinumad ja tõhusamad meetodid kasutajate andmete varastamiseks, mis kasutavad ära inimeste tähelepanematust ja teadmatust. Web eID autentimislahenduse laialdane kasutus muudab selle oluliseks sihtmärgiks sellistele rünnakutele. Pilt pildis õngitsusrünnak on võrlemisi vähetuntud rünnak, mida viiakse läbi brauseris. Tegemist on keeruka ründetaktikaga, mille käigus luuakse erinevate tehniliste vahendite abil võltsitud hüpikaken, mis imiteerib ehtsat brauseri või rakenduse akent, kuid on täielikult ründaja kontrolli all. Sarnast lähenemist on võimalik kasutada ka Web eID autentimislahenduse vastu. Lõputöö eesmärk on uurida pilt pildis rünnaku teostatavust ja tõhusust Web eID autentimisakna vastu, et hinnata selle õngitsusrünnakuga kaasnevat turvariski. Selleks loodi lõputöö raames veebirakendus, mis demonstreerib pilt pildis õngitsusrünnakut Web eID autentimisakna vastu eesmärgiga hankida kasutaja PIN1. Loodud kontseptsiooni tõenduse tõhususe hindamiseks viidi läbi kasutajauuring. Tulemused näitasid, et vaid kaks osalejat kümnest suutsid tuvastada õngitsusrünnaku. See tõestab, et pilt pildis õngitsusrünnak Web eID autentimisakna vastu võib olla väga tõhus.

Võtmesõnad: Web eID, pilt pildis õngitsusrünnak, kasutajaliides

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia

Contents

1. Introduction 6
2. Picture-in-picture attack 9
3. Authentication with Web eID 11
    3.1 Technologies used 11
    3.2 Authentication process with the Web eID 11
4. Analysis of the Web eID authentication window 14
    4.1 Structure and contents of the Web eID authentication window 14
    4.2 Functionality of the Web eID authentication window 15
    4.3 Web Browsers 17
    4.4 Operating systems 18
    4.5 Operating system themes 21
    4.6 Screen resolutions 23
    4.7 Scale 26
    4.8 Page zoom 28
    4.9 Conclusion 28
5. Implementation details 29
    5.1 Design of the web application 29
    5.2 Creation of the fake authentication window 30
        5.2.1 Detecting the OS 30
        5.2.2 Structure and style of the fake window 31
        5.2.3 Functionality of the fake window 32
        5.2.4 Personal data from the authentication certificate 33
        5.2.5 Detecting the OS theme 34
        5.2.6 Positioning the fake window 34
        5.2.7 Size of the fake window 36
    5.3 Results 36
    5.4 Conclusion 43
6. User study 44
    6.1 Participants 44
    6.2 Methodology 44
    6.3 Procedure 45
    6.4 Results 48

6.5 Conclusion 52
7. Countermeasures and mitigation strategies 53
7.1 Technical countermeasures 53
7.1.1 Smart card readers with PIN pad 53
7.1.2 Darkening background of Web eID window 54
7.1.3 Positioning the Web eID authentication crossing the line of death 54
7.1.4 Minimising the browser window 55
7.2 Raising awareness 55
7.3 Conclusion 57
8. Conclusion and future work 58
References 59
License 62

1. Introduction

Phishing attacks are among the most common and effective ways to steal personal information from users. These fast-evolving attacks exploit people's lack of knowledge and inattentiveness. One of the lesser-known phishing attacks is a picture-in-picture attack that uses image elements inside a website to imitate a pop-up window. This gives the user the impression that the activities occur in the legitimate window. Actually, the user remains on the malicious site, and the attacker will possess all the data entered in the fraudulent pop-up.

The Web eID authentication solution provides a secure and user-friendly way to authenticate using an ID card to access different web services in Estonia. During the authentication process, an authentication window is displayed for the user asking to enter PIN1 (see Figure 1). This window could become a target for a picture-in-picture attack. Upon a successful attack, the attacker could obtain the user's PIN1. This information alone is not enough to inflict damage on the user. However, paired with physical access to the ID card or to a compromised system where the victim's ID card is inserted, the attacker could authenticate as the victim in government and private sector services. This would give the attacker access to sensitive personal information.

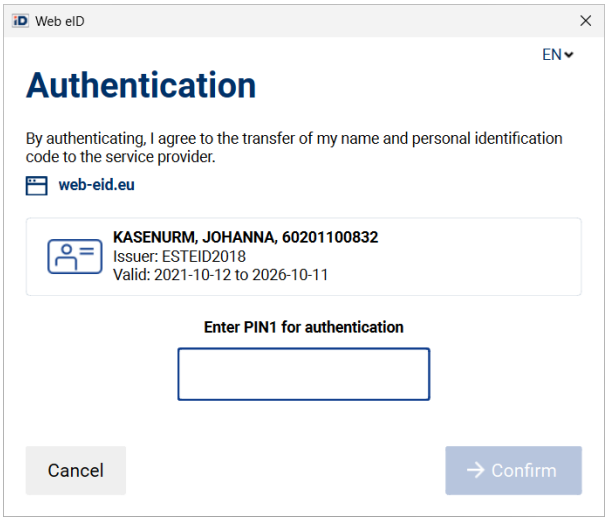


Figure 1. Web eID authentication window on Windows 11

In principle, a similar picture-in-picture attack could also be conducted against the digital signing functionality provided by Web eID, where the user is asked to enter the PIN2 (see Figure 2), thereby allowing the attacker to obtain the user's PIN2 as well. However, this work focuses on the Web eID authentication window only.

Figure 2. Web eID signing window on Windows 10

Estonian ID cards also support Transport Layer Security Client Certificate Authentication (TLS CCA), where authentication is performed as part of the TLS handshake to the server. This authentication method also involves an authentication window asking for PIN1 entry (see Figure 3). A similar picture-in-picture attack would likely also succeed against the TLS CCA window. Still, this thesis focuses on the Web eID solution recommended by the Information System Authority [1] and is believed to replace TLS CCA completely in the future.

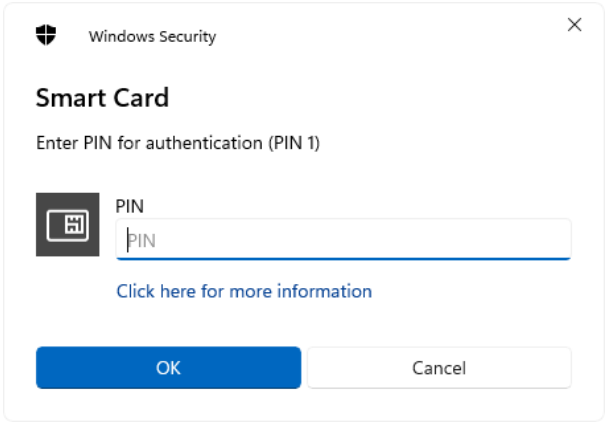


Figure 3. TLS CCA authentication window on Windows 10

The main goal of this thesis is to assess the feasibility and effectiveness of a picture-in-picture attack targeting the Web eID authentication window. To achieve this, a web application was developed to replicate the Web eID authentication window and demonstrate how it could be used in a fraudulent authentication process. In addition, a user study was conducted to evaluate

the efficiency of this attack. The remainder of the thesis describes the technical aspects of the implementation, the design, and the findings from the user study.

The thesis consists of eight sections. Section 2 describes the picture-in-picture and similar phishing attacks to provide the background knowledge needed. Section 3 introduces the authentication process with an ID card using Web eID to understand how a picture-in-picture attack could be conducted against it. Section 4 provides an extensive analysis of the Web eID authentication window necessary for successfully replicating it. Section 5 outlines the creation of the web application and the fake authentication window, and analyses the results. Section 6 describes the user study and its results. Section 7 discusses possible countermeasures to prevent and detect a picture-in-picture attack against the Web eID authentication window. Section 8 concludes the thesis and suggests potential improvements for the future.

2. Picture-in-picture attack

This section introduces the picture-in-picture attack and related work to provide the background knowledge needed to successfully demonstrate a picture-in-picture phishing attack against the Web eID authentication window.

The first picture-in-picture type of phishing attack can be found in a study by Jackson *et al*. [2]. They describe that the principle of a picture-in-picture attack lies in imitating the pop-up with the image inside the website, which gives the user the impression that the action occurs on another website or in another application. The study tested the effectiveness of different phishing methods in three test groups. Participants were tasked with correctly determining whether the submitted website was genuine or fake. For the picture-in-picture phishing attack, an image was displayed in the browser that appeared to be the window of another web page (see Figure 4). In addition, a homograph attack was used, for which domain names were visually very similar to those of genuine websites. The results showed that the picture-in-picture attacks were as effective as the homograph attacks. Only three out of 27 could identify a picture-in-picture phishing attack.

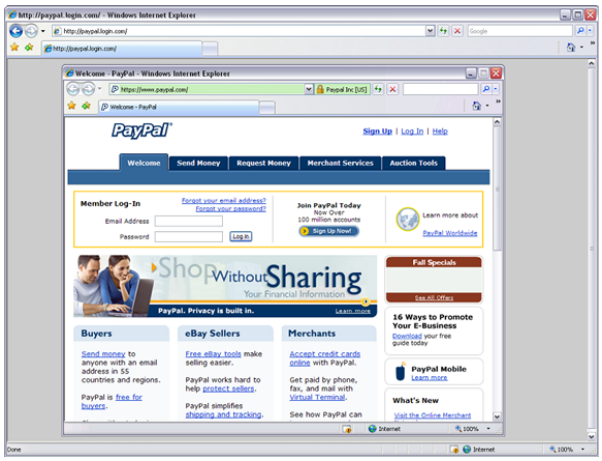


Figure 4. Picture-in-picture phishing attack. The outside window is real and the inside window is fake [2].

In some literature, the term browser-in-the-browser is used to refer to the attack described by Jackson *et al*. For example, in March 2022, ”mr.d0x” [3] introduced a similar attack, calling it a browser-in-the-browser attack, which takes advantage of the widespread use of the Single Sign On (SSO) authentication mechanism. SSO makes it possible to link the user’s account to a third-party service, simplifying the creation and login of a user account [4]. In his article, Grustniy explains that the browser-in-the-browser attack resembles the picture-in-the-picture

attack. However, instead of imitating another website, the login window of some well-known authentication services (e.g. Microsoft, Google or Apple) is imitated with an element inside the website (see Figure 5). Such a window looks identical to the original, but the usernames and passwords entered go directly to the attacker's server [5]. Additionally, Keski-Pukkila [6] describes a similar attack that spoofs the Windows credential pop-up with an element inside a website.

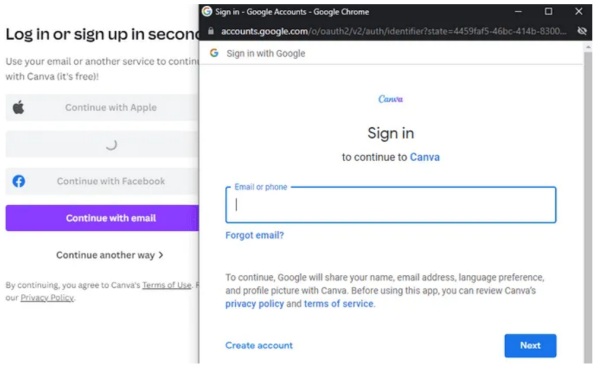


Figure 5. Browser-in-the-browser phishing attack [7]

In their study, Jackson *et al.* outline ways to detect a browser-in-the-browser attack. In particular, the fake window cannot be moved out of the original browser window or changed in size. Also, the fake windows close when minimised [2]. However, these detection methods require a great deal of attentiveness and awareness from the user.

A picture-in-picture attack can imitate elements like browser windows and SSO login windows inside a website to obtain sensitive information like passwords from the user. This type of attack can be difficult to detect since the fake window looks and behaves as expected.

3. Authentication with Web eID

This section provides the necessary background of the authentication process with an ID card using Web eID. It analyses the technologies needed for authentication with Web eID and discusses the details essential for replicating the authentication process in a picture-in-picture attack.

3.1 Technologies used

The Web eID solution provides a reliable and user-friendly method for web-based authentication with an ID card. To accomplish this, various technologies are used.

Web eID JavaScript library web-eid.js [8] integrates into the front end of a web application and handles communication with the Web eID browser extension. It provides secure authentication and digital signing with electronic ID smart cards to web applications.

The Web eID browser extension [9] manages the interaction with the Web eID native application installed on the user's computer, ensuring secure communication. It allows the web application to interact with the Web eID native application.

The Web eID native application [10] communicates directly with the smart card to perform cryptographic digital signing and authentication operations. The native application communicates the results to the Web eID browser extension.

3.2 Authentication process with the Web eID

The authentication process with Web eID follows these steps:

1. A user navigates to a website that supports ID card authentication and selects the option to log in with an ID card.
2. The web application calls the `webeid.authenticate()` function in JavaScript to initiate the authentication process.
3. The native application shows Web eID UI windows prompting the user to connect a card reader to the computer and insert an ID card into the reader (see Figures 6 and 7).

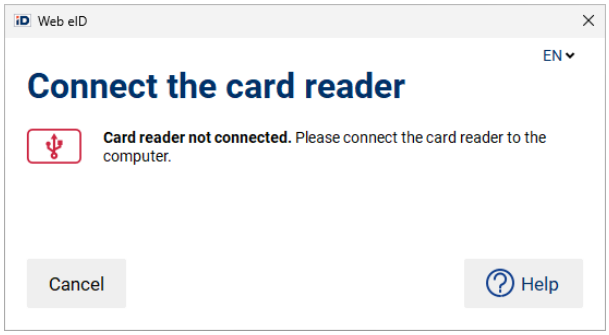


Figure 6. Web eID prompts the user to connect a card reader

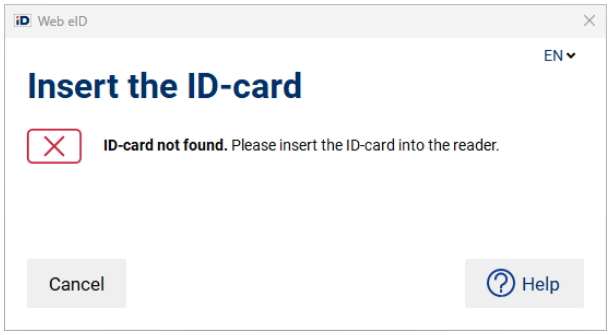


Figure 7. Web eID prompts the user to insert the ID card into the reader

4. After the card is inserted, the Web eID authentication window appears, prompting the user for PIN1 to authorise the authentication request (see Figure 8).

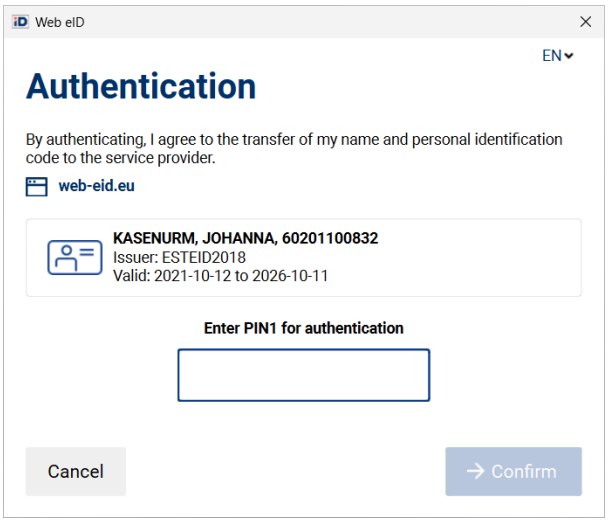


Figure 8. Web eID authentication window on Windows 11

5. After the user enters PIN1 and clicks on the "Confirm" button, the Web eID native application generates an authentication token by signing a challenge provided by the website using the user's authentication private key stored in the ID card [11]. The authentication token consists of the user's public key certificate, the digital signature, and the algorithm used for the digital signature.

6. The signed authentication token is returned via JavaScript as a return value of the `webeid.authenticate()` call, which the website's frontend can submit to the website's backend (server) for verification. The server verifies the token's validity by checking the signature and ensuring the certificate was issued by a trusted certificate authority [12].

7. Upon successfully verifying the authentication token, the website extracts the user's data, such as name and personal identification code, from the certificate. The user is then granted access to the website's services.

As can be seen, during the authentication process with the Web eID solution, an authentication window prompting the user to enter PIN1 is displayed. Since this window is shown on top of the open browser window, the website could perform a picture-in-picture attack by displaying a fake window instead of the legitimate Web eID authentication window. For this attack to succeed, the fake authentication window must look and behave identically to the legitimate window.

4. Analysis of the Web eID authentication window

The primary purpose of this section is to study how the appearance of the Web eID authentication window differs depending on the browser, screen resolution, operating system, operating system theme and scale factor used. This is necessary to understand the challenges of implementing a fake window. This section analyses the authentication window of Web eID version 2.6.0 (released on 08.10.2024). This includes its contents, structure, functionality, and appearance. The Web eID authentication window's appearance is observed in different browsers and operating systems, using different screen resolutions, operating system themes and scale factors.

4.1 Structure and contents of the Web eID authentication window

The Web eID authentication window consists of a title bar and main content area (see Figure 9). The title bar includes a close button and the title text "Web eID", accompanied by an ID icon in the top-right corner. Below the title bar, users can select their preferred language using a drop-down menu. This is followed by a consent notice and the domain name of the website the user is attempting to authenticate with. Next, the window displays information retrieved from the authentication certificate of the ID card connected to the computer. Below this section is the PIN1 input field. At the bottom of the window, 'Cancel' and 'Confirm' buttons are presented. The 'Confirm' button becomes active once the user has entered at least four digits into the PIN1 field.

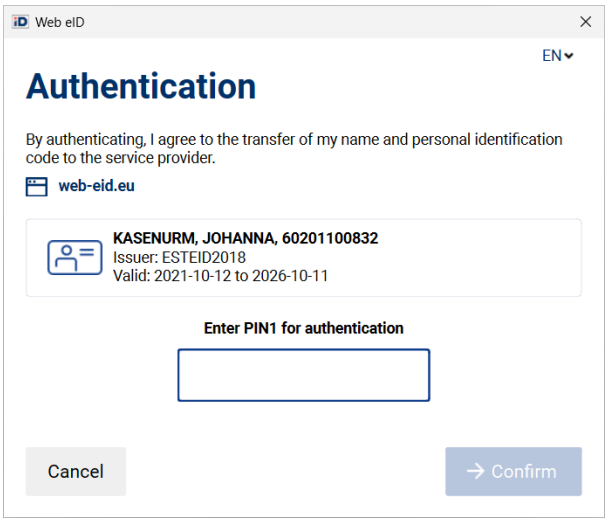


Figure 9. Web eID version 2.6.0 authentication window on Windows 11

As can be seen, the only variable information displayed in this window is the website's domain name (which depends on the website where the user is performing authentication) and the

information from the user's authentication certificate, which depends on the ID card inserted. The Web eID browser extension sets the domain name based on the website origin that made the call to `webeid.authenticate()`, while the personal information is taken from the authentication certificate stored on the user's ID card.

In the case of a fake window, a malicious website can easily display a domain name that matches the domain name of the website. However, to display the user's certificate information, the malicious website would first have to authenticate the user using the Web eID solution by calling the `webeid.authenticate()` function, which returns an authentication token containing the user's certificate. In the case of a targeted attack, the initial authentication process could be avoided as the malicious website could hardcode the personal information to match that of the victim's certificate obtained through other means. However, there is no guarantee that when the fake window is displayed to the victim, the victim has an ID card connected to the computer. In such a case, the victim might detect the attack. Therefore, performing a legitimate authentication process right before displaying the fake authentication window would be preferable to prevent the victim from detecting the attack.

4.2 Functionality of the Web eID authentication window

In addition to replicating the visual design of the Web eID authentication window, the fake window must also mimic its behaviour and functionality throughout the entire authentication process. Several key elements need to be reproduced. First, after the user clicks the login button, there is approximately a one-second delay before the Web eID user interface appears. This delay results from the time required to launch the native application process and render the UI window on the screen. When the native application starts, it initially displays a "Waiting for card" window accompanied by a loading spinner (see Figure 10). During this phase, the native app reads the user's authentication certificate from the connected ID card. However, because this intermediate window is typically visible for less than a second, its absence is unlikely to be noticed by the user. Therefore, replicating this window is outside the scope of the picture-in-picture proof-of-concept attack implemented in this work.

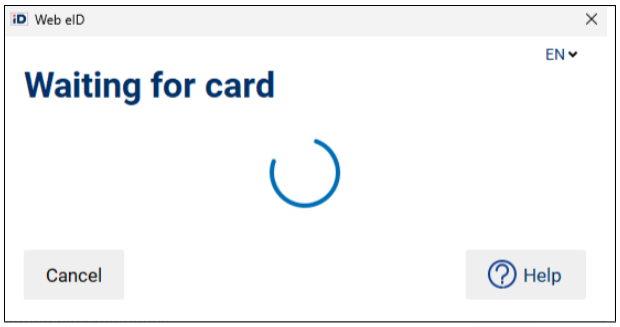


Figure 10. Web eID waiting for card window

When the Web eID authentication window opens, the PIN1 input field is automatically focused, showing a blinking cursor inside. The text inside the authentication window is not selectable, and the characters typed into the PIN1 input field are masked with asterisks (see Figure 11). Also, the input field does not accept any other characters besides digits. The confirm button remains disabled until the user has typed at least four digits. However, the maximum length of the input is twelve characters. Another detail is that the background colour changes for all the buttons when the user hovers the mouse over them. These visual effects can be replicated in a picture-in-picture attack using CSS (Cascading Style Sheets) and JavaScript.

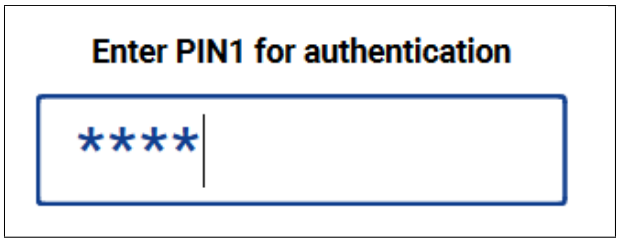
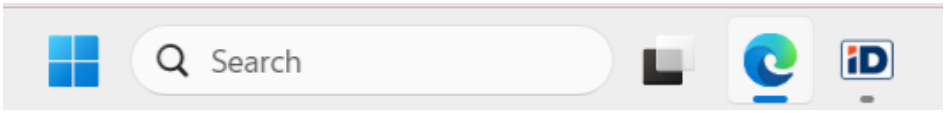


Figure 11. Characters typed into the PIN1 input field masked with asterisks

In addition, while the Web eID authentication window is open, the Web eID application icon is shown in the operating system's taskbar (see Figure 12). This icon is shown as the result of the Web eID native application process that starts when the authentication window is opened. In the case of a picture-in-picture attack, no Web eID application process is started. Therefore, replicating the Web eID application icon in the taskbar is impossible.



(a) Windows 11

(b) macOS Monterey

(c) Ubuntu 22.04

Figure 12. Web eID application icon in the taskbar of different OS

Additionally, the browser window loses focus when the Web eID authentication window appears. This is visually denoted by slightly changing the colour of the browser's title bar. In the case of a picture-in-picture attack, the browser would not lose focus. However, since the change in colour of the browser's title bar is very subtle (see Figure 13), the user is unlikely to notice that when a fake window is shown, the browser window does not lose focus.

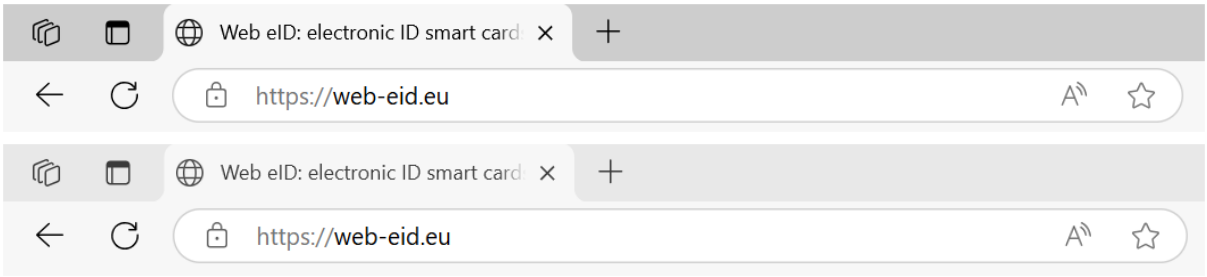


Figure 13. Browser window color when focused (top) and unfocused (bottom) on Windows 10

The following subsections analyse the design of the Web eID authentication window across different conditions, including browser type, screen resolution, operating system, OS theme, and scaling. Understanding these variations is essential for accurately replicating the Web eID authentication window in the implementation of the picture-in-picture attack.

4.3 Web Browsers

The four most common web browsers, Chrome, Safari, Edge, and Firefox [13], were tested to see whether the appearance of the Web eID authentication window changes depending on the browser used.

The observations showed that the browser used does not affect the appearance of the Web eID authentication window. Therefore, the web browser factor can be disregarded when performing a picture-in-picture attack targeting the Web eID authentication window.

4.4 Operating systems

It was decided to test how the appearance of the Web eID authentication window changes depending on the operating system. The selected operating systems were Windows 11, Windows 10, Ubuntu 22.04, Ubuntu 24.04 and macOS Monterey. As of March 2025, the most popular operating system for desktop computers in Estonia is Windows (83.32%), followed by OS X (10.89%), macOS (2.52%) and Linux (1.97%) [14]. The two most common versions for Windows are Windows 10 (54.2%) and Windows 11 (42.69%) [15]. The most popular distribution for Linux is Ubuntu, with version 22.04 being the most widely used version and version 24.04 being the latest LTS (Long Term Support) version [16].

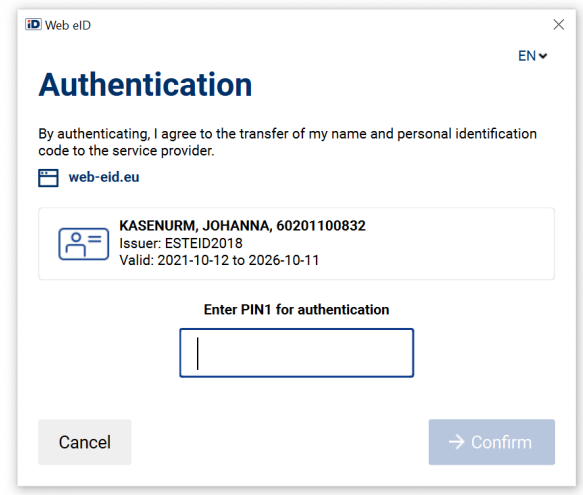
The comparison between the Web eID authentication windows in the selected operating systems is shown in Figure 14. As can be seen, the authentication window appears differently depending on the operating system used. Most differences lie in the title bar, while the window contents are essentially the same.

To summarise:

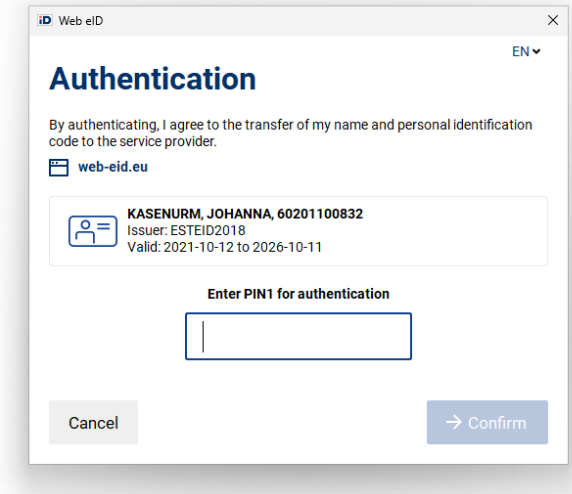
- On Windows OS, title bars have an icon, and the title is shown on the left, while on Ubuntu and macOS, the title bar has no icon and the title is centred.
- The window has rounded corners on macOS, but on Windows, the corners are sharp. On Ubuntu, the window's top corners are rounded and the bottom corners are sharp.
- The authentication window on Ubuntu 24.04 has minimise and maximise buttons at the top right corner next to the close button, while on Windows and Ubuntu 22.04, there are no such buttons. There are also minimise and maximise buttons on macOS, but they are in the top left corner.
- On macOS, the buttons in the title bar are represented by circles, while on Ubuntu, they are represented by circles with icons inside them. On Windows, the buttons are marked by just icons.
- The Web eID authentication windows on Ubuntu 24.04, Ubuntu 22.04 and macOS have a more distinctive line between the title bar and the rest of the window than on Windows.

- The authentication window on Ubuntu 24.04 has a thinner shadow than the authentication windows on other operating systems.
- On Windows 10, the Web eID authentication window does not have a distinctive border like on the other operating systems.
- On Windows 10, the eID logo in the title bar has rounded corners, but on Windows 11, the eID logo has sharp corners.

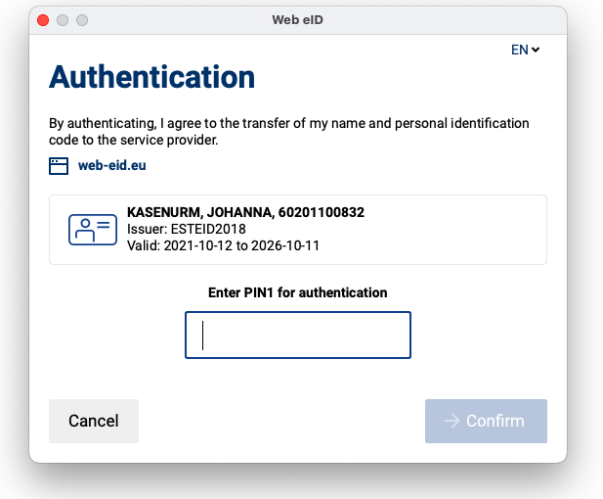
These observations show that the operating system affects the appearance of the Web eID authentication window the most. This means the attacker must be able to detect the victim's operating system and display a fake authentication window with a matching appearance. This complicates a picture-in-picture attack since the attacker must create multiple fake authentication windows for different operating systems. If the attacker fails to present the correct window, the victim could notice the differences, and the attack might fail. However, this heavily depends on the victim's attentiveness to detail and the level of suspiciousness when the Web eID window appears slightly different from usual.



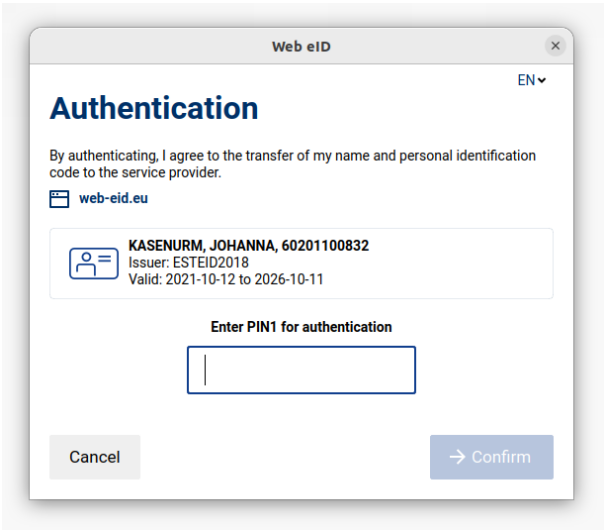
(a) Windows 10



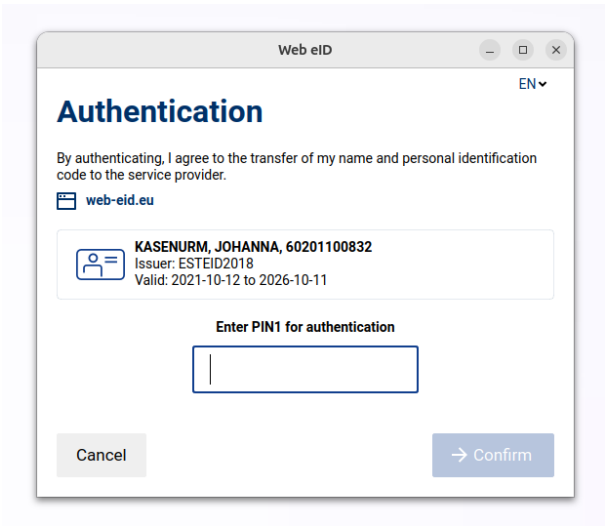
(b) Windows 11



(c) macOS Monterey



(d) Ubuntu 22.04



(e) Ubuntu 24.04

Figure 14. Web eID authentication windows on different operating systems

4.5 Operating system themes

All the selected operating systems give users a choice between light and dark themes, which makes the operating systems more personalisable. By default, the theme is set to light in the observed operating systems, and it is unknown how many users change it to dark. A survey conducted in 2019 [17] showed that about 82.7% of desktop and laptop users use the dark mode of their operating systems. However, since this survey was conducted on a test group of software developers, it might not accurately show the preferences of all users.

The operating system theme also affects the appearance of the Web eID authentication window. In the light theme, the authentication window has a light background with black or dark blue text (see Figures 15- 17). In the dark theme, the background is dark with white text. In the light theme, the ID card, window icon, and the PIN1 entry box's border are also dark blue. In the dark theme, the window icon is white, but the ID card icon and the border of the PIN1 entry box are blue. However, a different shade of blue is used in the dark theme than in the light theme. Also, the entered PIN1 is masked with dark blue asterisks in the light theme, but in the dark theme, the asterisks are white.

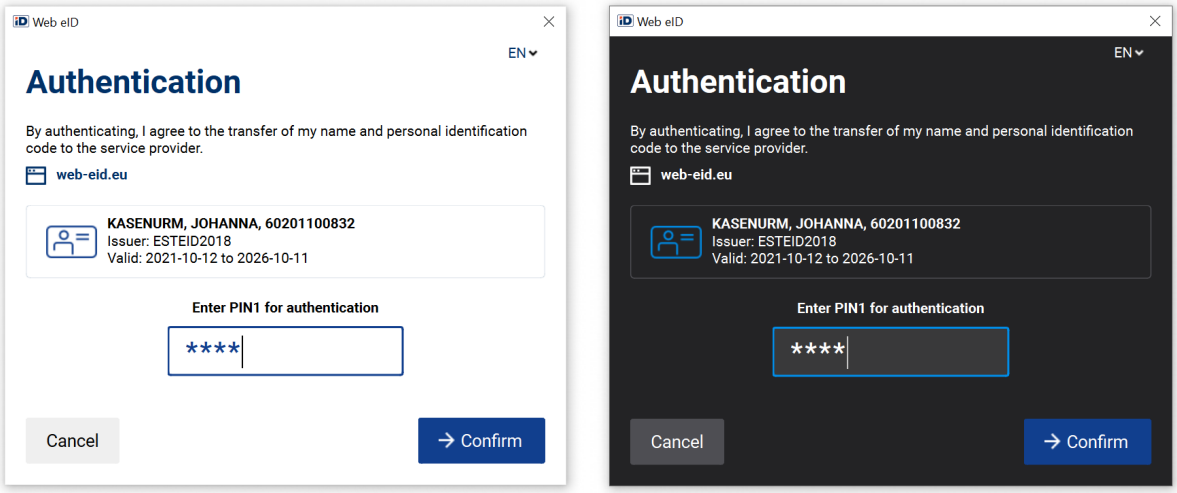


Figure 15. Web eID authentication window in light (left) and dark (right) theme on Windows 10

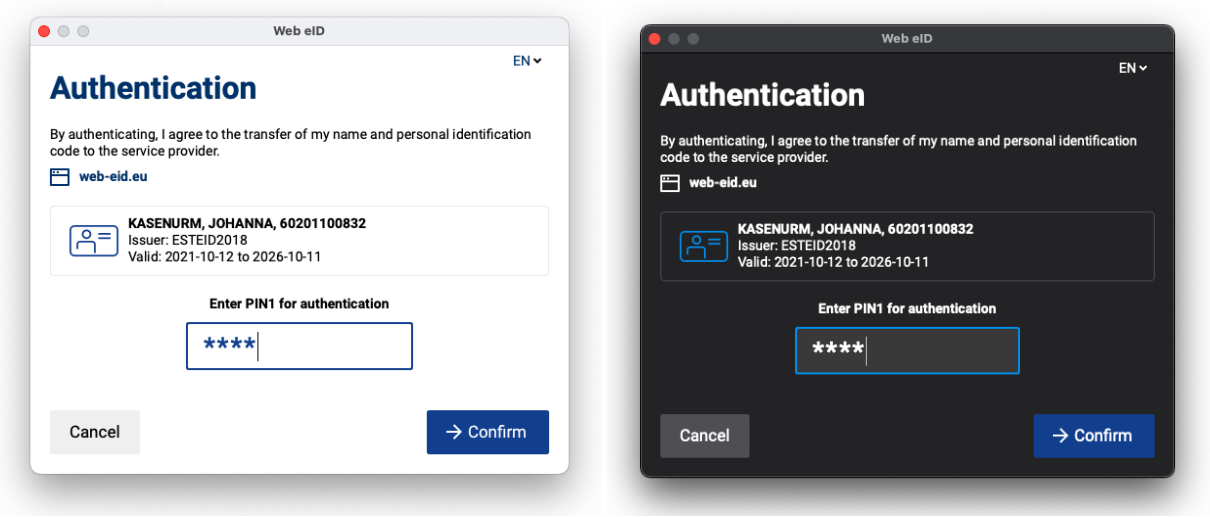


Figure 16. Web eID authentication window in light (left) and dark (right) theme on macOS Monterey

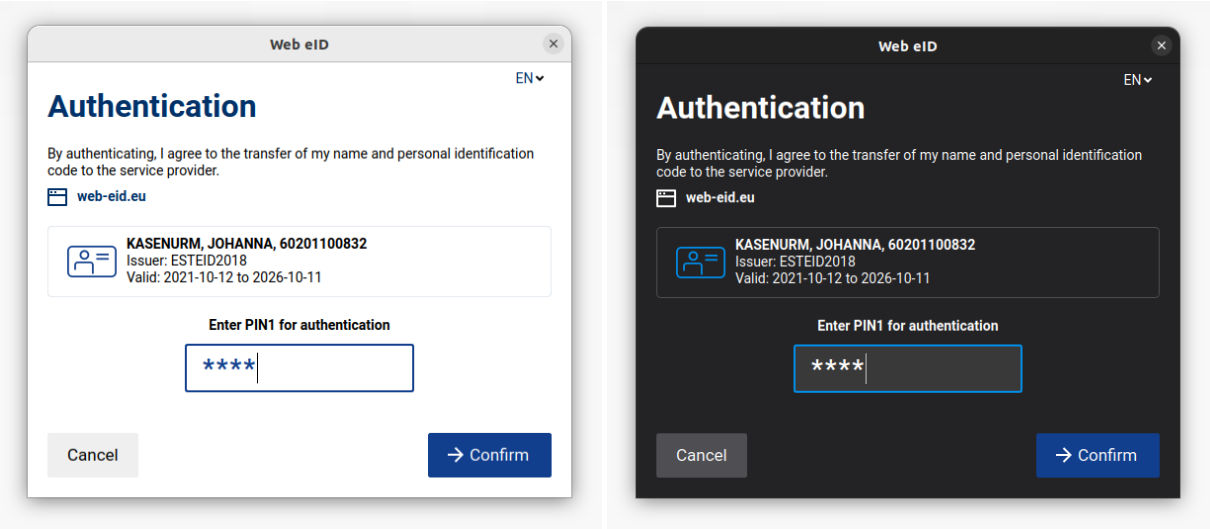


Figure 17. Web eID authentication window in light (left) and dark (right) theme on Ubuntu 22.04

There are also some differences between the light and dark themes that are not unanimous across all the selected operating systems. One example would be that on Windows 10 and Windows 11, the title bar of the Web eID authentication window has a light background regardless of the operating system theme used (see Figure 15). On other operating systems, the title bar's background in dark theme is dark, and the text is white or light grey (see Figures 16 and 17).

Also, on Windows 11, the title bar's background is slightly darker than on Windows 10 (see Figures 14a and 14b).

These observations show that the operating system theme is another variable that further complicates the picture-in-picture attack against the Web eID authentication window. Since the authentication window's appearance differs in light and dark themes, the attacker must be able to detect the operating system theme and adjust the colour scheme accordingly. If this is not done, the user might notice the difference between the legitimate and fake windows, leading to a potential detection of the attack.

4.6 Screen resolutions

Screen resolution refers to the number of pixels displayed horizontally and vertically on a screen, typically expressed as "width × height" [18]. Users can adjust their screen resolution through system settings, depending on their preferences or device capabilities. As of April 2025, the most commonly used screen resolution across all platforms in Estonia is 1920 × 1080 (see Figure 18).

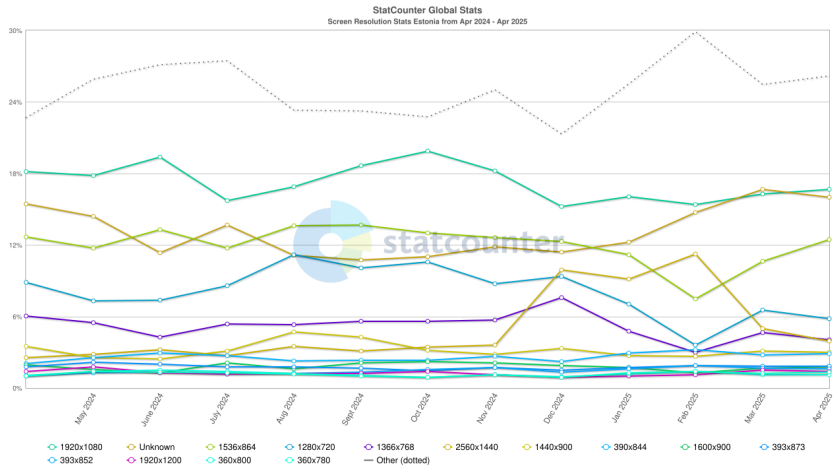


Figure 18. Screen resolution statistics across all platforms in Estonia from Apr 2024 - Apr 2025 [19]

The appearance of the Web eID authentication window was observed using four different screen resolutions in all the selected operating systems (see Table 1). This selection was made from the available screen resolutions with the aim of experimenting with a wide range of screen resolutions. When possible, the exact screen resolution was selected across all the operating systems to compare the observations in different operating systems.

The results show that screen resolution does not affect the appearance of the Web eID authentication window, meaning that its size in pixels does not change depending on the

Table 1. Screen resolutions used in analysis on the selected operating systems (in the order of largest to smallest).

macOS	Ubuntu 22.04 and 24.04	Windows 10	Windows 11
1440x900	1920x1080	1920x1080	1920x1080
1440x810	1680x1050	1680x1050	1920x904
1280x720	1280x720	1280x1024	1280x720
1024x768	1024x768	1024x768	1024x768

resolution. However, it can affect the window's vertical position on the screen. On macOS, the authentication window is positioned at the top of the screen, partly covering the top of the browser window where the address bar, tabs and security indicators are displayed (see Figure 19). However, on Ubuntu 24.04, Ubuntu 22.04, and Windows 10, the entire authentication window was placed below the top of the browser window when the screen resolution was 1920x1080 or 1680x1050 (see Figures 20, 21 and 22). Additionally, on Windows 10, the authentication window was displayed below the top of the browser with a screen resolution of 1280x1024. On Windows 11, the authentication window was placed under the top of the browser window when the screen resolutions were 1920x1080 and 1920x904. This is an important difference from the security perspective because the malicious website cannot draw anything outside its content area. The boundary between the website's content area and the browser's interface is known as the "line of death" [20]. This term highlights that a malicious website can potentially control anything below the line to execute attacks such as a picture-in-picture attack.

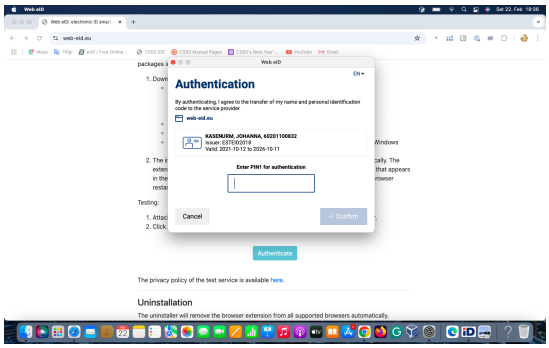


Figure 19. Web eID authentication window on macOS Monterey with screen resolution 1440x900

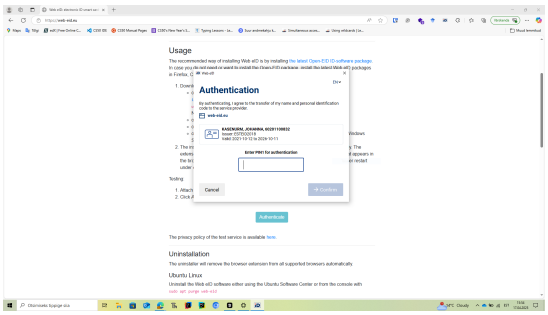


Figure 20. Web eID authentication window on Windows 10 with screen resolution 1920x1080

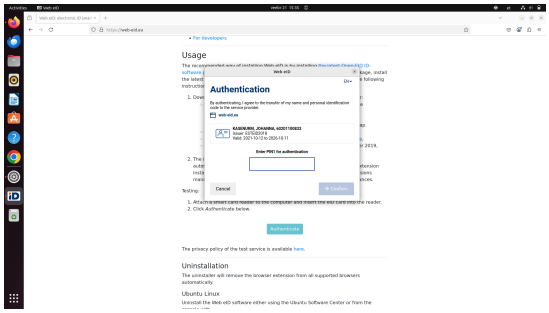


Figure 21. Web eID authentication window on Ubuntu 22.04 with screen resolution 1920x1080

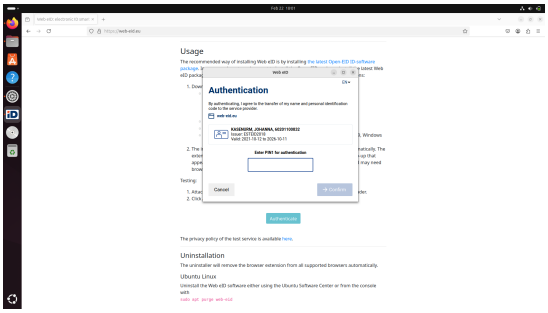


Figure 22. Web eID authentication window on Ubuntu 24.04 with screen resolution 1920x1080

The screen resolution did not influence the authentication window's horizontal position. The analysis showed that it was displayed slightly to the left of the screen (12 pixels) in all the tested screen resolutions on all operating systems (see Figure 23). However, the inclination to the left is not significant, and therefore, the authentication window for the user may appear to be centred.

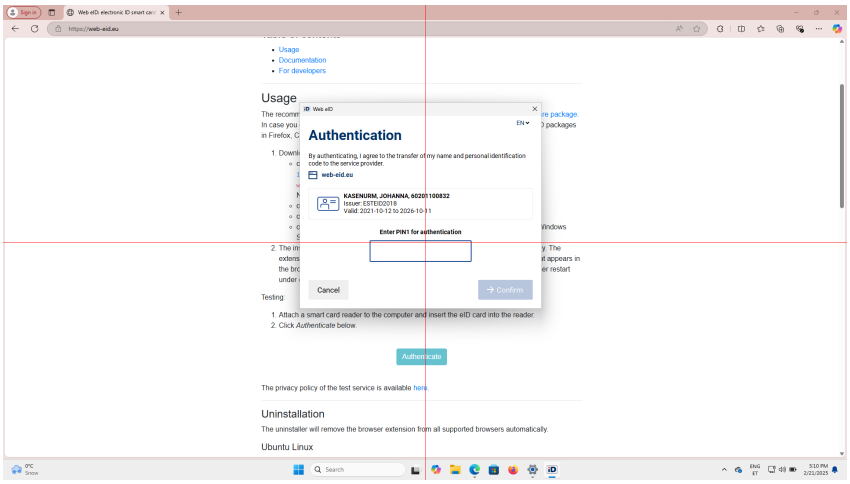


Figure 23. Web eID authentication window in relation to the centre of the screen on Windows 11 with screen resolution 1920x1080

The differences between the Web eID authentication window's position in different screen resolutions are an essential factor in conducting a picture-in-picture attack, since the attacker would need to match the position of the legitimate window. However, it is impossible to position the fake window above the line of death. This is a significant limitation that could enable the detection of the picture-in-picture attack if the user considers it suspicious that the fake window is not rendered in its usual place.

The size of the legitimate Web eID authentication window is 827 x 705 pixels. Hence, this is also the minimum browser content area size for the malicious website for the picture-in-picture attack to be successful. If the browser's content area is smaller, part of the fake authentication window will not be displayed, and the user could grow suspicious and detect the attack.

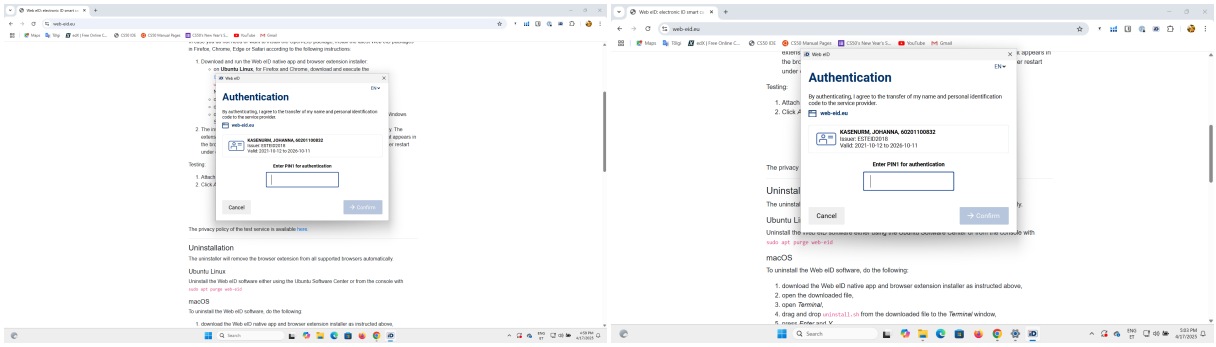
4.7 Scale

Screen zoom (scale) is another factor that affects the position and size of the Web eID authentication window. It adjusts the size of text, icons, and user interface elements to enhance readability. Scale is usually denoted with a percentage (e.g., 100%) and controls how many physical pixels are displayed in one visual pixel [21]. Thus, with a scale of 200%, two physical pixels are rendered as one visual pixel on a screen.

All the operating systems, except for macOS, allow users to choose between different screen zoom factors. Windows 10 and 11 present options of 100%, 125%, 150%, and 175%. On Windows 10, the default scale is 150%, and on Windows 11, it is 100%. On Ubuntu 22.04 and 24.04, the user could choose from 100% and 200%. In both versions, the default was 100%.

The effect of the applied scale was tested on all the operating systems that allowed changing the scale for a screen resolution of 1920x1080. This is the most common screen resolution in Estonia and was available in both versions of Windows and Ubuntu.

The observations showed that a bigger scale made the authentication window appear larger and positioned it more towards the upper part of the screen. In both versions of Windows, the authentication window was displayed above the line of death when the scale was 150% and 175% and below it when the scale was 100%. However, when the scale was 125%, the window's position differed on Windows 10 and 11. On Windows 11, the window appeared above the line of death; on Windows 10, it appeared below it (see Figures 24a- 24d). On Ubuntu 22.04 and 24.04, the Web eID authentication window was positioned below the line of death with scale 100% and crossed it with scale 200% (see Figures 24e and 24f). Similarly to the screen resolution, the scale did not affect the horizontal position of the authentication window.



(a) Windows 11 with scale 100%

(b) Windows 11 with scale 125%

(c) Windows 11 with scale 150%

(d) Windows 11 with scale 175%

(e) Ubuntu 22.04 with scale 100%

(f) ubuntu 22.04 with scale 200%

Figure 24. Web eID authentication windows on different operating systems with different scale factors.

Since the scaling affects both the legitimate Web eID authentication window and the fake window equally, no special handling is needed to adjust the size of the fake authentication window. However, the scale also changes the position of the original authentication window, which needs to be matched with a fake window. With some scaling factors, the Web eID authentication window is positioned above the line of death. Since the fake window can not be displayed outside the page content area, it is impossible to match the legitimate window's position entirely. This could enable the detection of the attack if the user notices the difference between the windows' positions. The fact that the scaling factor affects the position of the Web

eID authentication window differently in different operating systems is another complication that needs to be overcome.

4.8 Page zoom

In addition to the operating system scale, browsers also allow users to change the zoom factor of a page [22]. This factor adjusts the size of all the elements inside the page content area, but does not affect the operating system and applications. Therefore, while this setting does not affect the size of the original Web eID authentication window, it affects the size of the fake window that is rendered in the webpage, making its size different from the original.

Chrome, Safari, Edge, and Firefox allow users to change the default page zoom that affects all websites and set a custom page zoom for specific sites. If the victim has changed the default page zoom, then the malicious website and the fake authentication window will be affected. This will cause the sizes of the legitimate authentication window and the fake window to differ, leading to the victim potentially detecting the attack. However, a site-specific page zoom is likely not a concern for a picture-in-picture attack when a victim visits the malicious website for the first time, as the victim is unlikely to change the page zoom, assuming that the malicious website uses a readable font size.

While there might be methods for the malicious website to detect the page zoom and adjust the size of the fake window accordingly, this factor was disregarded in the proof-of-concept implementation of a picture-in-picture attack.

4.9 Conclusion

The observations analysed in this section show that the appearance and position of the Web eID authentication window can differ depending on the operating system, operating system theme, display resolution and scale used. Some differences are more noticeable than others. This complicates the picture-in-picture attack since the appearance of the fake authentication window has to be tailored to each operating system and operating system theme. An attentive user could notice if there are visual changes in the appearance of the authentication window.

5. Implementation details

The practical part of this thesis aimed to demonstrate a picture-in-picture spoofing attack against the Web eID authentication window. For this purpose, a web application was created that replicates the legitimate Web eID authentication window. For this purpose, three versions of the fake window were created, replicating the Web eID authentication window of Windows 11, macOS Monterey, and Ubuntu 22.04. To increase the success potential of the attack, the fake windows needed to be as similar as possible to the original Web eID authentication windows. Various techniques were used to achieve this. The purpose of this section is to provide an overview of how the web application and the fake authentication window were created and used to demonstrate the picture-in-picture spoofing attack.

5.1 Design of the web application

As a result, a web application was created and is available for testing at https://login.websec.ee/ (see Figure 25). The login page contains a login box with three buttons: ”Smart-ID”, ”Mobile-ID”, and ”ID-card”. The ”ID-card” button starts the legitimate authentication process with Web eID when clicked on for the first time. After successful authentication, the ”ID-card” button displays a fake authentication window when clicked on.

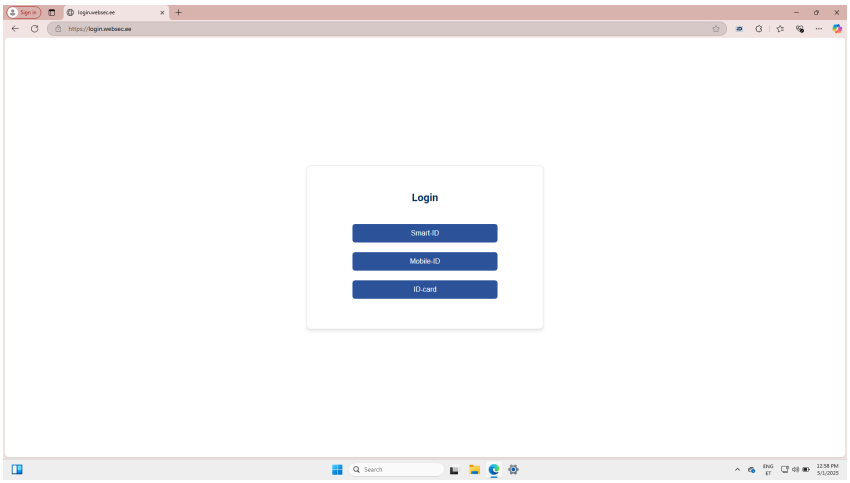


Figure 25. Login page of the proof-of-concept web application

The testing mode of the created web application is shown in Figure 26. It allows for testing fake and legitimate authentication windows and comparing them side-by-side. The page contains four buttons: “Real window”, “Fake window on Windows”, “Fake window on macOS”, and “Fake window on Linux”. Clicking on the first button initiates the legitimate authentication process with Web eID. The rest of the buttons display fake authentication windows for different

operating systems. The testing page also contains a button that allows changing the theme of the fake authentication windows, a button for clearing data stored in the browser's local storage, and information about the detected operating system and theme. For testing the fake authentication windows, an ID card is not necessary, in which case, the data in the fake window's certificate information field will be filled with placeholder data. After the real authentication process, the placeholder data is replaced with the information from the user's authentication certificate.

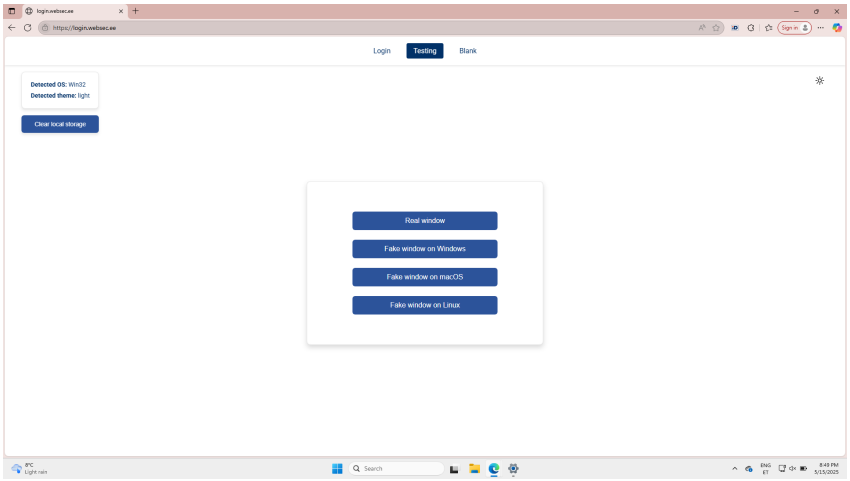


Figure 26. Testing page of the proof-of-concept web application on Windows 11

The proof-of-concept code was implemented using HTML (HyperText Markup Language), CSS and JavaScript scripting language. No server-side code was used because, for a proof-of-concept attack, no data was meant to leave the user's computer. It also made the application faster and easier to host.

5.2 Creation of the fake authentication window

The Web eID authentication window has many aspects that need to be replicated by the fake authentication window. The following subsections aim to describe how each aspect of the legitimate window was replicated.

5.2.1 Detecting the OS

Since the design of the Web eID authentication window differs depending on the operating system, the proof-of-concept implementation needs to detect the user's operating system to display a fake window with the correct appearance.

JavaScript allows the user's operating system to be extracted through the JavaScript's `navigator.platform` property. This property provides information on the platform the

user's browser is running on [23]. The possible values returned by this attribute could be, for example, "Win32," "Linux x86_64," and "MacIntel".

Even though it is possible to detect the user's operating system by analysing this value, it does not contain any information about the specific operating system version. Another way to obtain the user's operating system would be to inspect the Navigator's `userAgent` property, which can be obtained by accessing `navigator.userAgent` in JavaScript. It contains information about the application, operating system, vendor, and version of the requesting user agent [24]. Although the `userAgent` property contains information about the version of the operating system, it is unreliable since this information could be wrong or missing. For example, for Windows 10 and 11, the version from the `userAgent` property in Edge was "Windows NT 10.0". For Ubuntu 22.04 and 24.04, the operating system version in Chrome was not specified.

WhatIsMyBrowser v3 API [25] provides a way to determine the user's operating system and version more reliably. It uses HTTP Headers known as Client Hints in addition to User Agents to provide more accurate information about the user's operating system. However, using this API requires server-side implementation to send requests to the API endpoints. Since this web application was designed without relying on server-side code, it was decided to use the Navigator's `platform` property to extract the user's operating system without differentiating between OS versions.

5.2.2 Structure and style of the fake window

Since it is possible to detect whether the victim uses Windows, macOS, or Linux, the proof-of-concept implementation of a picture-in-picture attack can be personalised for these operating systems. Therefore, three authentication windows were created for different operating systems. For Windows, the Web eID authentication window on Windows 11 was chosen to be replicated, since Windows 10 will be discontinued on October 14, 2025 [26]. As for Linux, the authentication window on Ubuntu 22.04 is replicated because Ubuntu 22.04 is currently the most widely used version. However, the differences between Windows and Ubuntu subversions are not significant. Therefore, the user is unlikely to notice if an authentication window for a different version of the same OS is used.

HTML was used to create and display the elements of the authentication window. It also allowed structure and hierarchy to be added to the created elements. However, HTML does not focus on appearance. Therefore, CSS was used to add style to the existing HTML elements and make the fake window visually similar to the legitimate Web eID authentication windows.

The Web eID authentication window also contains several icons. To match the visual appearance as closely as possible, this implementation uses the same icons as the Web eID native app [27]. On Windows 10 and 11, the authentication window's title bar also contains the ID icon. For the fake window, the icon was obtained from the official website [28].

The fake authentication window must also use the same font as the original window to make the text visually identical. The legitimate Web eID authentication window uses the Roboto font [29]. Therefore, the fake authentication window also uses the Roboto font obtained from the Web eID native app's GitHub page [30].

As for font size, the fake window uses various font sizes for different elements. The title "Authentication" is an `h1` element with a font size of 32px. The user consent notice is a paragraph with a font size set to 14px. The "Cancel" and "Confirm" buttons have a font size of 17px, while the language dropdown button has a font size of 13px and the languages in the dropdown menu have a font size of 16px. However, determining the font size of the PIN1 in the input area was not this straightforward. To match the size of the asterisks in the legitimate window, the font size of the masked PIN1 was initially set to 36px. This, however, made the input field and the cursor appear larger than in the original authentication window. With adjustments to the padding of the input field, it was possible to achieve the correct size of the input field, but the cursor remained too long and was not centred. To counter this, JavaScript was used to adjust the padding and change the font size of PIN1 to 28px when the input field's value is empty and to 36px when the user starts typing. This ensures the cursor is centred and the correct size when the fake window appears.

5.2.3 Functionality of the fake window

In addition to appearance, the fake authentication must have the same UI functionality as the legitimate Web eID authentication window. For this purpose, JavaScript was used. This technology allowed the replication of various Web eID authentication window functionalities.

One of the most important uses for JavaScript was to make the buttons functional. For this purpose, event listeners were added to detect when the user clicks the buttons and respond accordingly. The languages button in the top right corner of the authentication window shows and hides the languages options when clicked on. The close and cancel buttons cancel the authentication process and hide the authentication window. The purpose of the confirm button was to finish the authentication process and close the fake window after a successful attack.

JavaScript was also used to disable the confirm button until the user had typed at least four characters in the input field.

JavaScript was also used to manage the PIN1 input field. Since the password-type input field available in HTML automatically masks the input with dots instead of asterisks, a text-type input field was used. This allows the user to input text but displays it as plaintext in the input field. JavaScript was used to mask the input with asterisks and prevent non-digit input. JavaScript was also used to automatically focus the PIN1 input field when the fake authentication window is shown.

Another characteristic of the legitimate Web eID authentication window is draggability. This functionality was implemented based on the code shared by P. Keski-Pukkila [6]. Keski-Pukkila's code provided draggability for the fake authentication window by using JavaScript to detect mouse drag movements and calculate the new position of the authentication window. However, dragging the fake window over the browser's title bar, like the legitimate authentication window, is impossible. When the fake window is dragged outside the page content area, the part that goes outside the content area is hidden from view (see Figure 27).

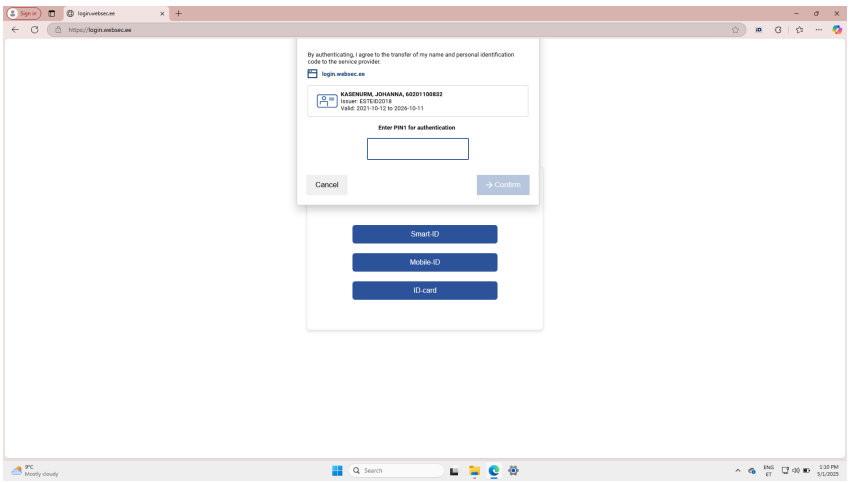


Figure 27. Fake authentication window dragged outside the page content area

5.2.4 Personal data from the authentication certificate

The original Web eID authentication window also contains information from the authentication certificate stored on the user's ID card. This includes the user's full name and personal identification number, the issuer of the certificate, and the validity dates. In order to obtain the user's certificate from the ID card, the official authentication process with Web eID must be completed before a fake window can be displayed.

The Web eID JavaScript library web-eid.js was used to initiate the legitimate authentication process by calling the authenticate function. Upon successful authentication, the function returns an authentication token that contains the user's certificate. This base64-encoded X.509 authentication certificate contains the information displayed in the Web eID authentication window.

The RSA-Sign [31] JavaScript library was used to parse the X.509 authentication certificate and extract the necessary information. Afterwards, JavaScript was used to convert the obtained information into a suitable format and display it in the fake authentication window. The extracted information was stored in the browser's local storage to make the personal data from the user's certificate available to the application even after the page is reloaded.

5.2.5 Detecting the OS theme

The previous analysis showed that the operating system theme is another factor affecting the appearance of the Web eID authentication window. Therefore, the fake authentication window's appearance must also change depending on the theme used.

The `prefers-color-scheme` CSS media feature can be used to detect the theme of the user's operating system [32]. This feature was used in JavaScript to detect the current theme and change the colours of backgrounds and texts accordingly. The window and ID card icons displayed in the authentication window are also replaced according to the user's theme. Additionally, event listeners were added in JavaScript to make the necessary updates if the user changes the theme in the settings or clicks on the button in the top right corner of the testing page.

5.2.6 Positioning the fake window

In addition to appearance, the position where the fake authentication window is displayed on the screen is also essential. Since the legitimate authentication process with Web eID must be completed before the fake window can be displayed, the user will see the Web eID authentication window shortly before the fraudulent window. If the position of the fake window differs significantly from the original, the user could grow suspicious, and the attack could fail.

The position of the Web eID authentication window is influenced by the screen resolution, scale factor, and operating system. Since it was impossible to reliably extract the operating system version using JavaScript, this factor was left out when calculating the position of the

fake authentication window. Instead, a proposed solution relies solely on screen resolution and scale factor and aims to calculate a position that can be used in all operating systems.

The width and height of the screen can be obtained by accessing the properties `window.screen.width` and `window.screen.height` via JavaScript. However, these values do not consider the location and size of the visible page content area. This poses no issue if the user's browser window occupies the entire screen. However, these values remain unchanged if the user's browser window only partly covers the screen. Therefore, using these values to calculate the position of the fake authentication window would cause it to be rendered in the same place regardless of whether the page content area covers this position. A better alternative would be to use `window.innerWidth` and `window.innerHeight`, which return the width and height of the visible page content area. This ensures the fake window is not positioned outside the page content area even when the browser window takes up only part of the screen.

The display scale is another factor that influences the position of the Web eID authentication window. This can be obtained from `window.devicePixelRatio`. The values returned by `window.innerWidth` and `window.innerHeight` already consider the display scale and do not need further adjustments. Thus, it is sufficient to use the extracted width and height of the page content area and the scale factor when calculating the horizontal and vertical position of the fake authentication window.

The observations showed that the horizontal position of the Web eID authentication window does not depend on the screen resolution and scale factor. The authentication window is always displayed slightly to the left of the screen. Therefore, the number of pixels to the left of the fake window is calculated by first centring the window and then moving it slightly to the left by distracting twelve pixels using the following code:

```
const width = window.innerWidth;
function calculateLeft(ui) {
    const uiWidth = ui.offsetWidth;
    return Math.max(0, Math.floor((width - uiWidth) / 2) - 12);
}
```

Determining the vertical position of the fake authentication window proved to be more challenging since it varies greatly depending on the scale factor and screen resolution. In

addition, the legitimate Web eID authentication window is sometimes positioned above the line of death, but for the fake window, this is not possible. Therefore, it is impossible to match the original window's position entirely. However, the aim is to devise a way to display the fake window as close to the legitimate window's position as possible. To accomplish this, a function was created that calculates how many pixels should be from the line of death to the top of the fake window:

```
const height = window.innerHeight;
const zoomFactor = window.devicePixelRatio;

function calculateTop() {
    const exponent = Math.round((zoomFactor - 1) / 0.25);
    const percentage = 0.11 / Math.pow(10, exponent);

    let pixels = Math.floor(height * percentage) - 8;
    pixels = Math.max(pixels, 10);

    return pixels;
}
```

As a result of the positioning algorithm, the vertical position of the fake window changes depending on the scale factor. As the scale increases, the fake window is positioned closer to the line of death, but never comes in contact with it. The algorithm ensures that at least 10 pixels remain between the line of death and the top of the fake authentication window at all times. This decision was made to make the position of the fake authentication window seem more natural.

5.2.7 Size of the fake window

The size of the fake authentication window is not adjusted by the web application and remains unchanged unless influenced by the operating system.

5.3 Results

Side-by-side comparisons of fake and original windows are available in Figures 28, 29, and 30. The fake authentication windows are visually similar to the Web eID authentication windows on Windows 11, macOS Monterey, and Ubuntu 22.04. The fake windows have all the elements from the original window and replicate the most critical functionalities like the PIN1 entry field, cancel and confirm buttons. They also read and display the user's personal information from the

ID card. The only visual difference between the fake and legitimate authentication windows is that the fake window's height is slightly bigger than the legitimate window's. Also, in the fake window, there is a small gap between the chevron down icon and the text "EN" in the language dropdown button, but in the legitimate authentication window, there is no gap. This is due to the fact that the chevron down icon is displayed as an SVG (Scalable Vector Graphics) element that does not enable positioning other elements directly next to the chevron without altering its size. Additionally, the text size in the title bar is slightly smaller in the fake authentication window than in the legitimate window.

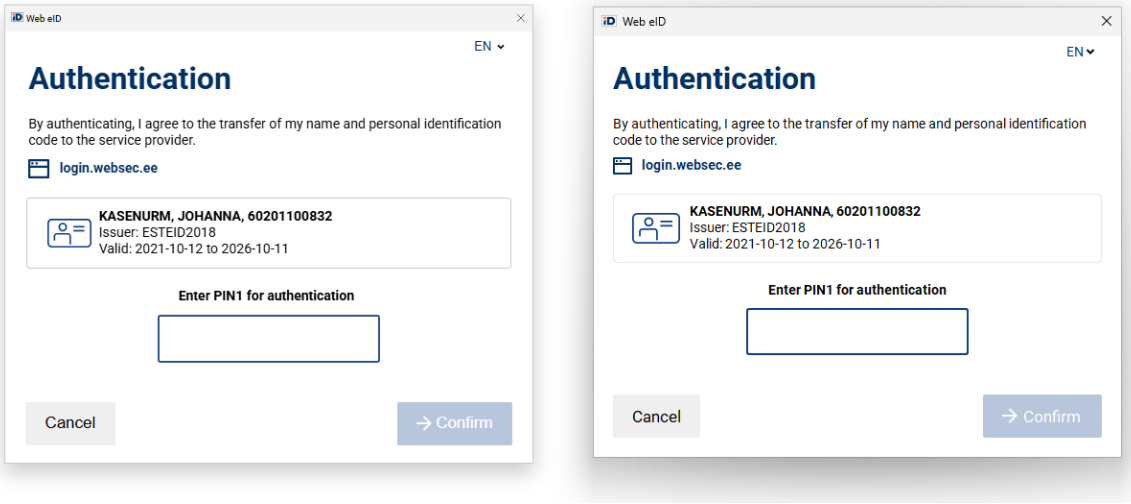


Figure 28. The fake (left) and legitimate (right) Web eID authentication windows on Windows 11 using the light theme

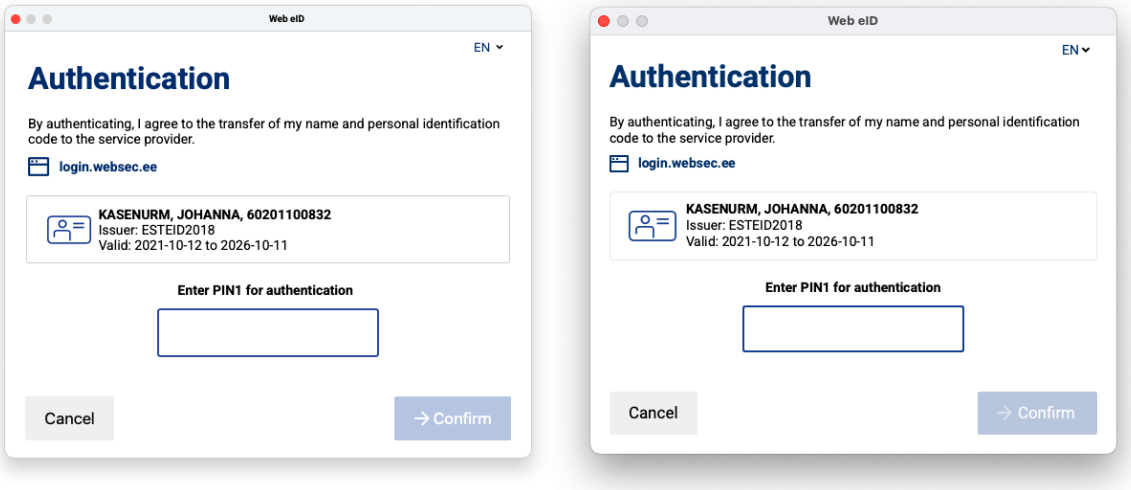


Figure 29. The fake (left) and legitimate (right) Web eID authentication windows on macOS Monterey using the light theme

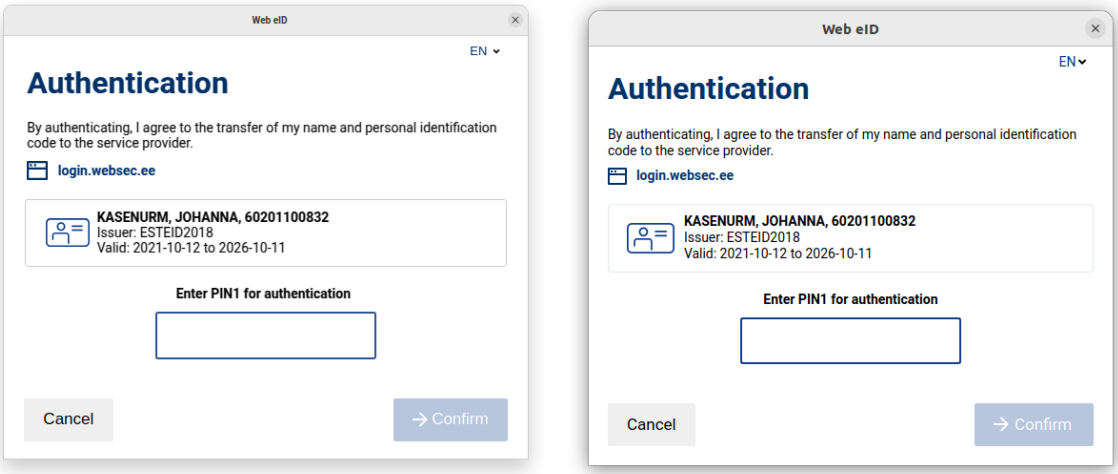


Figure 30. The fake (left) and legitimate (right) Web eID authentication windows on Ubuntu 22.04 using the light theme

The appearance of the fake authentication windows also changes depending on the user's operating system theme, similar to the legitimate Web eID authentication window (see Figures 31- 33). This remains true even if the user changes the operating system theme during authentication.

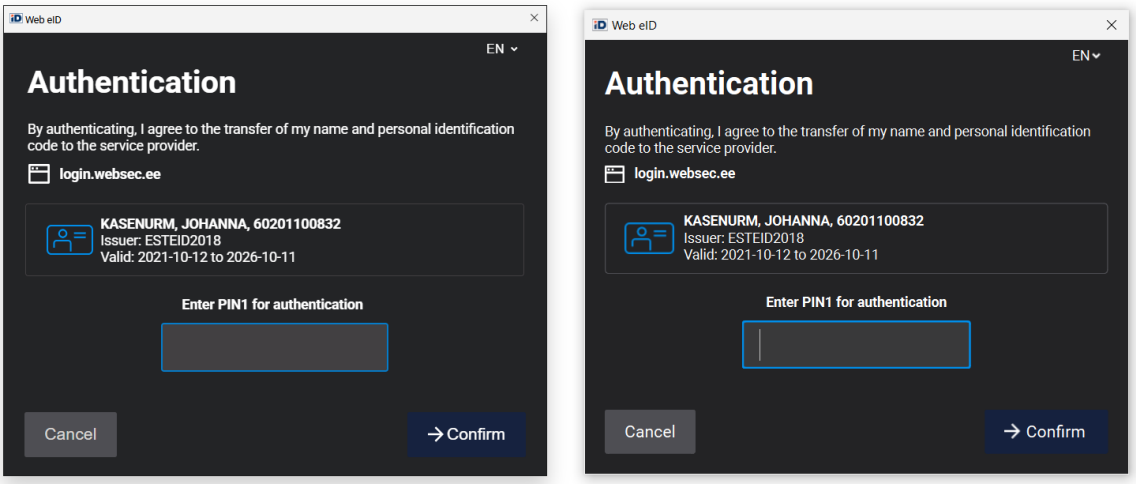


Figure 31. The fake (left) and legitimate (right) Web eID authentication windows on Windows 11 using the dark theme

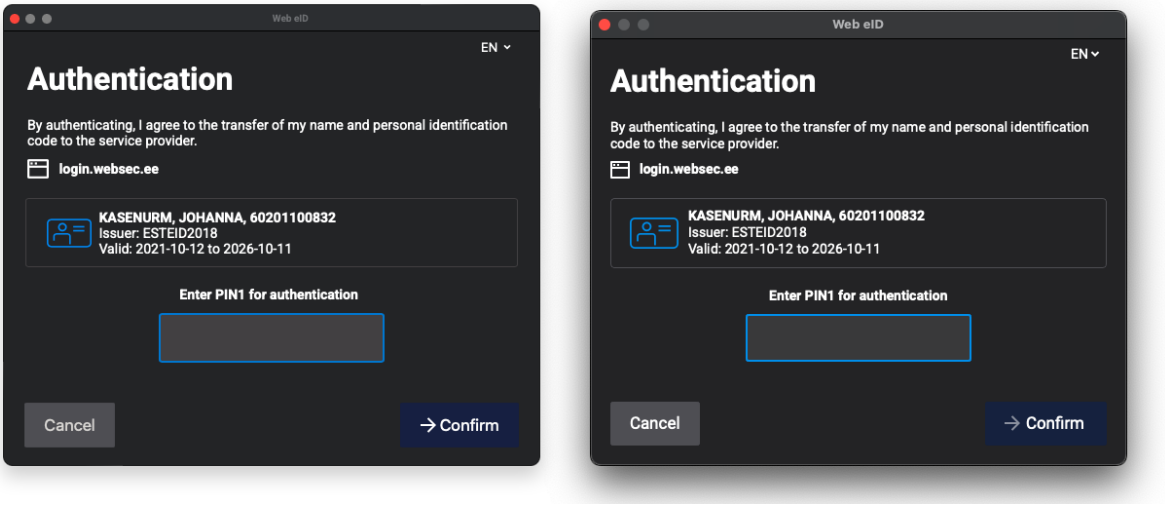


Figure 32. The fake (left) and legitimate (right) Web eID authentication windows on macOS Monterey using the dark theme

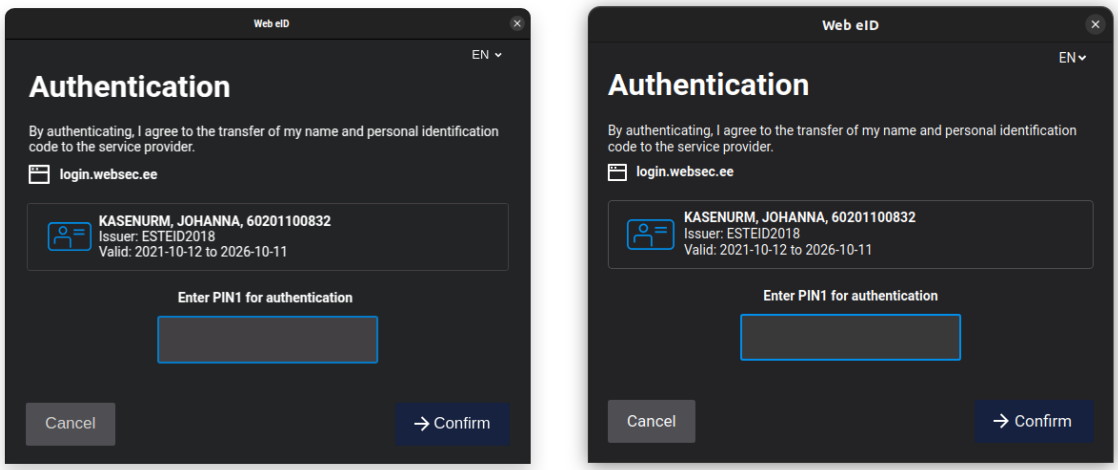


Figure 33. The fake (left) and legitimate (right) Web eID authentication windows on Ubuntu 22.04 using the dark theme

Even though the fake authentication windows are visually similar to the real ones, some limitations would enable the attack detection. One of the most significant limitations is that the fake windows cannot be moved outside the website, and there is no process in the taskbar. Also, the browser window is focused while the fake window is focused. This can not be changed and could indicate an attack if the user is observant.

Another limitation is that only the authentication windows for Windows 11, macOS Monterey, and Ubuntu 22.04 operating systems are replicated. If the user has a different operating system,

then the visual differences between authentication windows in various operating systems could give away the attack. However, the appearance of the Web eID authentication window is relatively similar regardless of the operating systems, so there is a chance that the user would not notice the difference.

In addition, detecting the operating system theme used is not always reliable. This is because the browser can override the operating system theme, and in this case, the web application would detect the browser's theme. If the user's browser and operating system themes do not match, the fake windows will appear different from the legitimate authentication window.

The original Web eID authentication window also has a language button allowing users to switch to a different language. This button is also present in the fake windows. It displays a list of available languages, such as those in the legitimate authentication window, but does not allow changing the language. The fake windows are always displayed in English. Still, the legitimate authentication window is also hardcoded to appear in English in this web application by passing the parameter "en" to the `webeid.authenticate()` function. Therefore, there is no noticeable difference unless the user tries to switch to a different language. However, in case the user has previously manually changed the language of the legitimate Web eID authentication window, the original window appears in the user's preferred language regardless of the parameter passed by the web application. In this case, the legitimate and fake windows will be in different languages, which could indicate an attack.

In addition, the position of the fake authentication windows does not entirely match that of the legitimate authentication window. With some screen resolutions and scales, the original Web eID authentication window is displayed above the line of death (see Figure 34). However, it is impossible to position the fake authentication window above the line of death, and the user could notice this difference (see Figure 35).

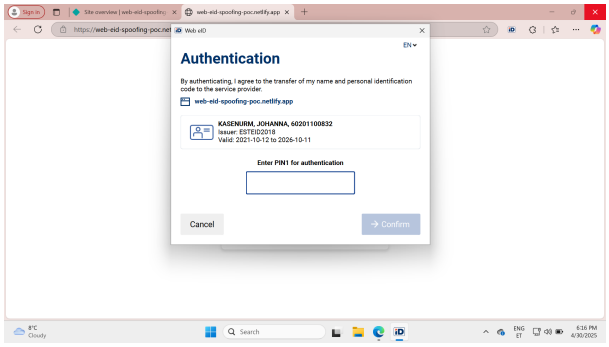


Figure 34. Web eID authentication window in the PoC application on Windows 11 with screen resolution 1920x1080 and scale 150%

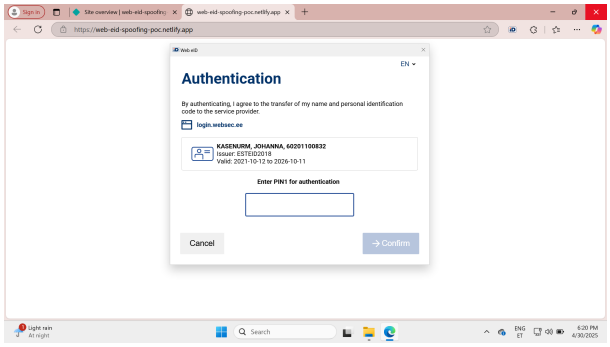


Figure 35. Fake authentication window in the PoC application on Windows 11 with screen resolution 1920x1080 and scale 150%

Furthermore, the proposed solution for calculating the location of the fake authentication window is best suited for a screen resolution of 1920x1080. It does not give accurate results in smaller resolutions (see Figures 36 and 37). However, the screen resolution 1920x1080 is the most common resolution in Estonia, and chances are that the user has this resolution. Besides, it is not clear how much users expect the Web eID authentication window to always appear in the same location on the screen.

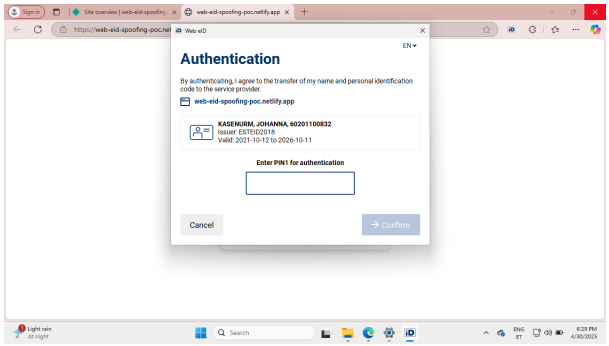


Figure 36. Web eID authentication window in the PoC application on Windows 11 with screen resolution 1280x720 and scale 100%

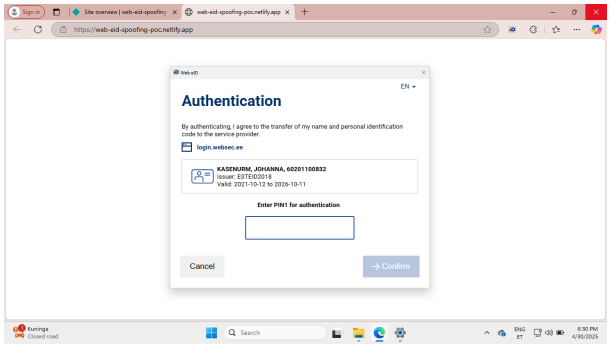


Figure 37. Fake authentication window in the PoC application on Windows 11 with screen resolution 1280x720 and zoom 100%

Additionally, it is impossible to display the fake authentication windows in a position with no page content area. This limitation becomes an issue when the user's browser window does not cover the entire screen. In this case, the legitimate Web eID authentication window's location remains unchanged (see Figure 38), but the fake authentication windows are displayed inside the page content area (see Figure 39). Furthermore, if the page content area cannot fit the entire

fake authentication window, the fake authentication window will be displayed to the extent that fits inside the page content area (see Figure 40).

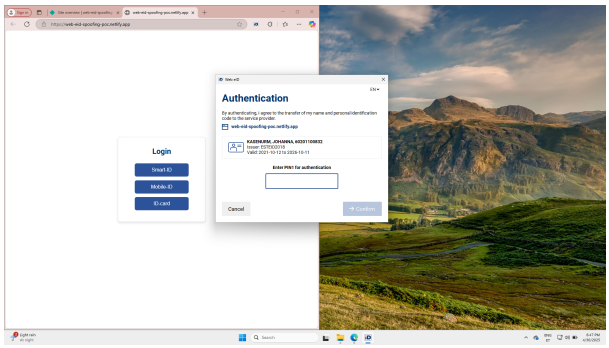


Figure 38. Web eID authentication window in the PoC application on Windows 11 with screen resolution 1920x1080 and scale 100%

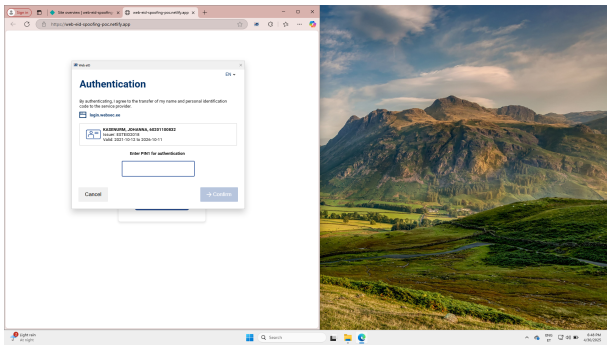


Figure 39. Fake authentication window in the PoC application on Windows 11 with screen resolution 1920x1080 and scale 100%

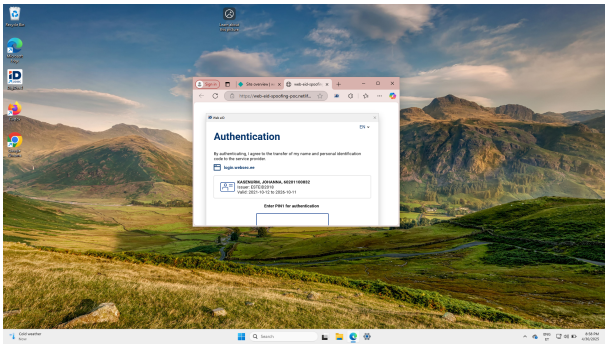


Figure 40. Fake authentication window on Windows 11 in a small browser window

Also, the function to calculate the location of the fake window was created based on the position of the legitimate authentication window on Windows 11 and is not designed to consider other operating systems. However, the differences in the location of the Web eID authentication window on various operating systems are not drastic. Therefore, there is a possibility that the proposed calculation could be used in other operating systems as well to carry out a successful picture-in-picture attack.

In addition, page zoom may also affect the picture-in-picture attack. If the user has changed the default zoom or changes the page zoom for the malicious website, then the legitimate and fake windows' sizes will differ (see Figures 41 and 42). Depending on how much the user has changed the page zoom, they could notice this difference and detect the attack.

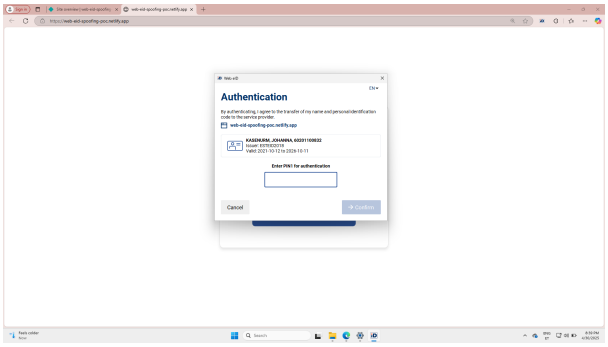


Figure 41. Web eID authentication window on Windows 11 with page zoom 150%

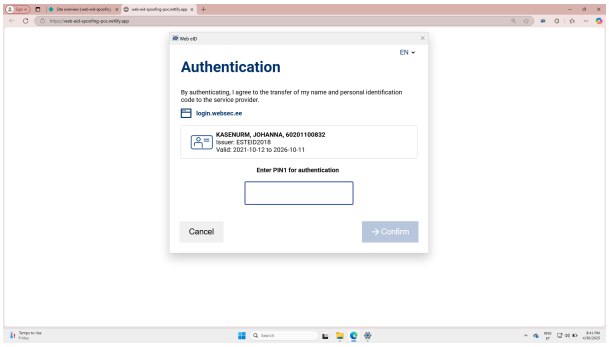


Figure 42. Fake authentication window on Windows 11 with page zoom 150%

5.4 Conclusion

The developed proof-of-concept implementation of a picture-in-picture attack against the Web eID authentication window achieves a quite close resemblance to the legitimate authentication window. The application uses the legitimate authentication process to obtain the user's ID card certificate information. Then, according to the user's operating system and theme, one of six fake windows is displayed on the next authentication attempt. The fake windows are visually similar to the Web eID authentication windows in Windows 11, macOS Monterey, and Ubuntu 22.04, but they have slight limitations that could enable the detection of the attack.

6. User study

In order to evaluate the effectiveness of the implemented proof-of-concept picture-in-picture attack against the Web eID authentication window, a user study was conducted. This section aims to describe the user study and analyse the results.

6.1 Participants

The user study had 10 participants who could all be considered experienced IT specialists, some even having an MSc degree in Cyber Security. The user study specifically targeted people with technical knowledge and who are well aware of common phishing attacks. This enabled testing the attack against a group of people expected to have the best ability to detect potential phishing attacks. Therefore, the results provide the best case performance in detecting the implemented picture-in-picture attack. In the case of average users with no technical background, a much worse performance in the ability to detect the attack should be expected.

6.2 Methodology

For the results of the study to be accurate, the precise objective of the study couldn't be revealed beforehand. The potential participants were sent an email invitation with general information describing the user study and how to prepare for it (see Listing 1).

Listing 1. Email invitation to participants

```
Subject: Invitation to participate in a user study
Dear [participant's name],

We invite you to participate in a study to evaluate the usability of the Web eID authentication solution.
This study aims to analyse human aspects involved when performing authentication with an ID card using the
Web eID solution. The results of this study will be used in a pseudonymised form in a bachelor's thesis.

The study takes place in Delta Centre (Narva mnt 18, 50090, Tartu) in room 3090 and takes approximately
15 minutes. It involves authenticating with an ID card using the Web eID solution in a test environment
and providing feedback on your experience. Some basic demographics, such as education and age group,
will also be collected.

What you'll need to bring:
- your ID card
- your computer
- an ID card reader (can be provided)

Before participating, please make sure that you can authenticate with an ID card on websites. To test this,
visit the https://eesti.ee/ webpage and log in with your ID card.
If you don't feel comfortable entering PIN1 in the presence of others, we recommend temporarily changing
your PIN1 code. The instructions can be found at this website:
https://www.id.ee/en/article/changing-id-card-pin-codes-and-puk-code/.

Best regards,
Johanna Kasenurm
```

The user study was conducted in a controlled in-person environment, one participant at a time. The room was reserved for the study, and only the thesis author, supervisor and the participant were present. This ensured that possible distractions were minimised.

The participants were asked to bring and use their personal computers for the user study. This ensured that the operating system, screen resolution, operating system theme, and other variables affecting the appearance of the authentication window were not known beforehand, making the experiment more similar to a real-life scenario.

In order to prepare for the study, the participants were asked to complete the ID card authentication process on the `eesti.ee` website. This ensured that the participants had correctly installed and set up the Web eID native application and the Web eID browser extension and had access to their PIN1. Since the participants were observed during the Web eID authentication process, they were advised beforehand to temporarily change their PIN1 for the study. This suggestion was made to ensure that the user study does not endanger the privacy of the participant's PIN1 and is not a security threat to the participants.

The user study required the participants to use their computers to authenticate with an ID card in a specially developed web application. In essence, the process consisted of participants performing real authentication first and then authenticating again after a simulated technical glitch. However, on the second time, a fake window was displayed to the participants. During the process, the participants were asked questions to collect information relevant to the study.

6.3 Procedure

All participants went through the following user study scenario:

1. Upon arriving at the room, the participant was instructed to start up their computer and was asked how frequently they use Web eID for authentication to websites.
2. Then, the participant was asked to navigate to the test environment available at `https://login.websec.ee/`, which displayed a mock login page (see Figure 43). At this point, the participant was asked whether it is safe to authenticate to random websites and the possible risks of authenticating to a malicious website with an ID card.

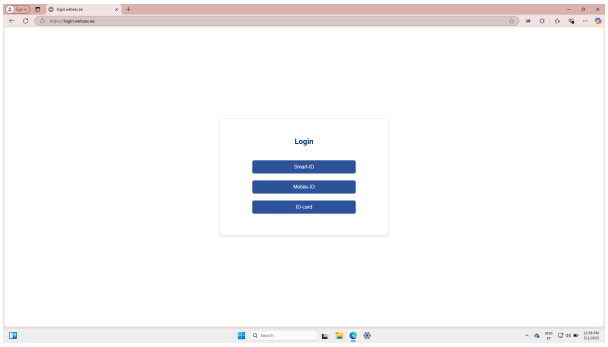


Figure 43. Login page of the implemented web application on Windows 11

3. Then the participant was asked to click on the button “ID-card”. The legitimate Web eID authentication window appeared (see Figure 44), and the participant was asked to highlight any important elements shown in the window from a security perspective. This ensured that the participant was forced to think about the security aspects of the window and had sufficient time to observe the legitimate window.

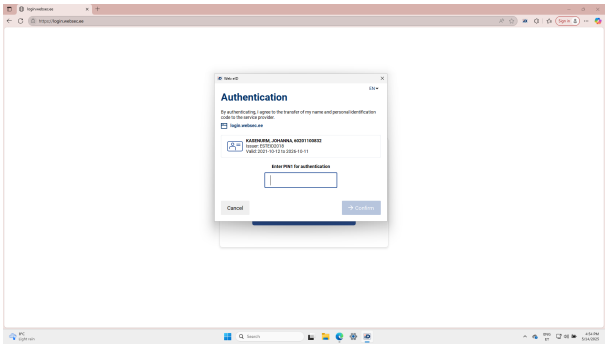


Figure 44. Web eID authentication window on the login page of the proof-of-concept web application on Windows 11

4. Next, the participant was asked to log into the website. Upon entering PIN1 in the legitimate window and clicking the “Confirm” button, the website displayed a spinning bar simulating a loading process (see Figure 45).

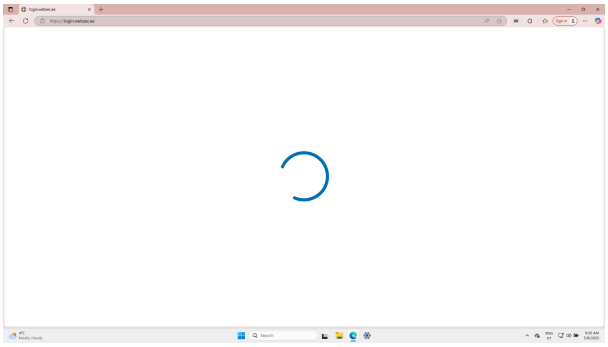


Figure 45. Simulated page loading in the implemented web application on Windows 11

5. After approximately 10 seconds of no progress, the persons conducting the study acted as if this was an unexpected technical glitch and advised the participant to reload the page and try logging in again.
6. If the user entered PIN1 into the fake window, the web application displayed an alert containing the first digit of the entered PIN1 (see Figure 46).

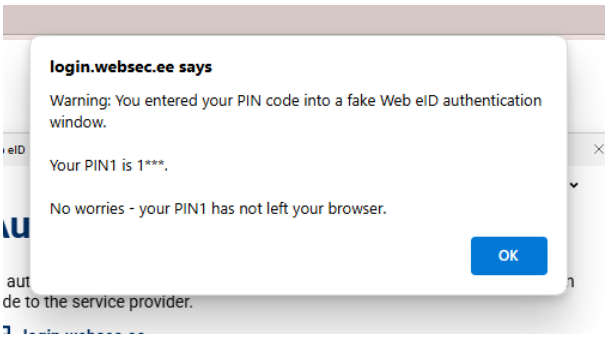


Figure 46. Warning alert in the implemented web application on Windows 11

7. If the participant detected the picture-in-picture attack and did not proceed with the authentication, the participant was asked to explain what enabled the detection of the spoofed window.
8. The participant was asked to access the website's testing mode and compare the real window to the fake window side-by-side. Visual differences were noted in the participant's data sheet.
9. The participant was asked to answer the following questions:
   (a) When authenticating for the second time, did you notice anything unusual?
   (b) What is the operating system of your computer?

(c) What is the theme of your operating system?

(d) What is the theme of your browser?

(e) What is the screen resolution of your computer?

(f) What is the screen scale factor of your computer?

(g) How confident are you in your ability to recognise phishing attacks?

(h) What is your gender?

(i) What is your age?

(j) What is your education level?

10. The participant was shown the filled-out participant data sheet and asked to confirm the data's correctness and give permission to use it in the thesis. The participant's name was not recorded in the participant data sheet.

11. After answering the clarifying questions, all the participants were explained the concepts of a picture-in-picture attack. They were also informed on how to detect a picture-in-picture attack and the differences between the legitimate Web eID authentication window and the fake authentication window in this study. This was done to raise awareness of picture-in-picture attacks and help participants detect them in the future.

12. At the end of the user study, the participants were thanked for their participation in the study and given a chocolate, which was perceived as a positive surprise by the participants.

6.4 Results

Only 2 participants out of 10 detected the attack and refused to enter PIN1 into the fake Web eID authentication window.

Table 2 shows the main demographic characteristics of the participants, their computer configuration, and whether the participant detected the attack. As can be seen, the list of participants covers different age groups, genders, and computer configurations.

Table 2. General information about the participants of the user study.

No.	Detected	OS	Resolution/scale	Theme	Web eID usage	Detection confidence	Gender	Age	Degree
1	No	Windows 11	1920x1080	light	Several times a year	Somewhat confident	F	45+	MSc in Cyber Security
2	No	Windows 11	1920x1200/125%	dark	Several times a year	Somewhat confident	F	18-24	MSc in Computer Science (studying)
3	No	Windows 11	1920x1080/150%	light	Monthly	Somewhat confident	M	18-24	MSc in Computer Science (studying)
4	No	Windows 10	1920x1080/150%	dark	Once a year	Confident	M	25-34	PhD in Computer Science
5	No	Ubuntu 24.04	1920x1080	dark	Daily	Very confident	M	45+	MSc in Cyber Security (studying)
6	**Yes**	Arch Linux	1920x1080	dark	Several times a year	Somewhat confident	M	25-34	MSc in Cyber Security
7	No	Fedora 41	1920x1080	light	Once a year	Confident	M	18-24	BSc in Computer Science (studying)
8	No	Ubuntu 24.04	1920x1080	light	Several times a year	Very confident	M	35-44	PhD in Computer Science
9	**Yes**	Ubuntu 24.04	1920x1080	dark	Once a year	Confident	M	35-44	BSc in Information Technology (studying)
10	No	Ubuntu 22.04	1920x1080	light	Once a year	Confident	M	25-34	MSc in Computer Science

The participants used different versions of Windows and Linux, the two most common versions being Windows 11 and Ubuntu 24.04. The implemented web application correctly detected the participants' operating systems in all cases, displaying the corresponding fake authentication window for Windows or Linux. However, none of the participants used macOS, so the fake window for macOS remained untested.

Half of the participants used the dark theme, and half used the light theme. In all cases, the participants' operating system theme matched that of the browser. This allowed the web application to detect the participant's OS theme correctly, displaying the fake authentication window in the same theme as the original Web eID authentication window.

For most participants, the screen resolution was 1920x1080, and the scale was 100%. For one participant, the screen resolution was 1920x1200, and for three participants, the scale was either 125% or 150%. None of the participants changed the page zoom and used the default browser page zoom of 100%. Therefore, there were no considerable differences in size between the fake and legitimate authentication windows.

The user study showed that only two participants (Participants 6 and 9) could detect the picture-in-picture attack against the Web eID authentication window. In both cases, the participants used a Linux operating system with a screen resolution of 1920x1080 in a dark theme.

Participant 6 used Arch Linux and the Tiling Window Manager Sway with a custom colour scheme. The customisation rendered all windows with sharp corners and also caused all windows (including the legitimate Web eID authentication window) to open in full screen by default. Since the fake window did not open in full screen and did not have sharp corners, the participant immediately detected the attack. After a side-by-side comparison of the real and fake windows, it was observed that in the legitimate window, the title "Web eID" in the title bar was aligned to the left, and there was no close button. In the fake authentication window, however, the title "Web eID" was aligned to the centre of the title bar, and there was a close button in the top right corner.

Participant 9 used Ubuntu 24.04 with some customisation, which caused the legitimate window to always open in the top left corner of the screen. According to the participant, the mismatched position of the fake window and the fact that the title text on the fake authentication window was smaller than on the real window were the main indications of a picture-in-picture attack. After side-by-side comparison of the real and fake windows, it was observed that contrary to the

real window, the fake window did not have minimise and maximise buttons, and the corners of the fake window were rounded.

It is important to emphasise that Participant 9 expected exactly this type of attack to be executed in the user study. When asked to describe what elements should be inspected in the Web eID window from a security perspective, the participant pointed out that it should be verified whether the authentication window is a real application window by moving the window outside the browser. Therefore, it is very likely that even if there were no visual differences between the fake and legitimate windows, the participant would have detected the attack anyway. Other participants appeared not to expect a Web eID window spoofing attack during the user study.

The rest of the participants were unable to detect the picture-in-picture attack and entered their PIN1 in the fake authentication window. However, there were some visual differences between the fake and legitimate authentication windows that went unnoticed:

- In the case of Participants 1, 2, 3, 4 and 7, there were slight differences between the title bar colours.
- In the case of Participants 2, 3 and 9, the corners of the fake and legitimate authentication windows differed.
- In the case of Participant 5, the legitimate authentication window appeared in Estonian, while the fake authentication window appeared in English, but the participant did not notice the difference.
- In the case of Participant 7, the legitimate authentication window appeared in the top left corner of the screen, but the fake authentication window was displayed more in the centre. According to the participant's explanation, the participant noticed the change but did not suspect an attack.

During the user study, tiny visual differences between the real and fake windows were observed that were not observable when testing. For example, the colour of the title bar on Windows 11 can differ. For Participant 1, the title bar was light grey, but for Participant 2, it had a dark background similar to Ubuntu and macOS.

All participants had previously used the Web eID solution to authenticate on websites, though most reported infrequent usage. Specifically, four participants stated they use Web eID once a

year, and another four use it several times a year. One participant uses it monthly, while one uses it daily.

When asked about the security of authenticating on potentially malicious websites, 8 out of 10 participants believed it to be unsafe, noting that such websites could potentially capture PIN codes or personal data. Participant 7 additionally noted the risk of phishing through the use of a fake authentication window.

Regarding the security indicators shown in the authentication window, 9 out of 10 participants said they would check the origin. Participants 7 and 9 further mentioned that the window should be able to cross the browser's line of death, and Participant 3 emphasised the importance of seeing a separate process in the operating system's taskbar, yet failed to check this when the fake window appeared.

Regarding phishing awareness, four participants reported being somewhat confident in their ability to detect phishing attacks, four felt confident, and two felt very confident.

6.5 Conclusion

The results of the user study show that only two out of ten participants could detect the phishing attack. This proves that even people with strong IT skills who are aware of possible phishing attacks can fall victim to a picture-in-picture attack against the Web eID authentication window. It is reasonable to expect that users with no technical knowledge would be even less able to detect this type of phishing attack. The results also indicate that the appearance of the fake window does not necessarily have to match the real window's appearance exactly. However, since the test group for this study was relatively small, the study could not provide a comprehensive answer to the extent to which visual differences of the fake window affect the success of the attack. For example, there were no cases where a fake window of another operating system or in a different theme was displayed. To gain more insights into these aspects, the study could be repeated on a larger test group, introducing various levels of visual differences between the fake and legitimate Web eID authentication windows.

7. Countermeasures and mitigation strategies

The results of the user study showed that a picture-in-picture attack against the Web eID authentication window can be difficult to detect. Several countermeasures and mitigation strategies are proposed in this section to help prevent this attack.

7.1 Technical countermeasures

Several technical measures can be implemented in the Web eID solution to help users more easily detect spoofing of Web eID windows.

7.1.1 Smart card readers with PIN pad

Using a smart card reader with a PIN pad is an effective measure to mitigate a picture-in-picture attack against the Web eID authentication window. These smart card readers allow users to enter the PIN1 in the built-in pad and send it directly to the ID card [33]. Instead of the Web eID authentication window discussed in this thesis, a slightly different authentication window is displayed to the user, in which there is no PIN input field (see Figure 47). A malicious website could still display a fake window asking to enter a PIN, but this would be unusual, and the user would be able to detect the attack. However, it cannot be excluded that an inexperienced user may not notice the use of a different authentication window or see it as a cause for alarm and may still fall victim to the attack.

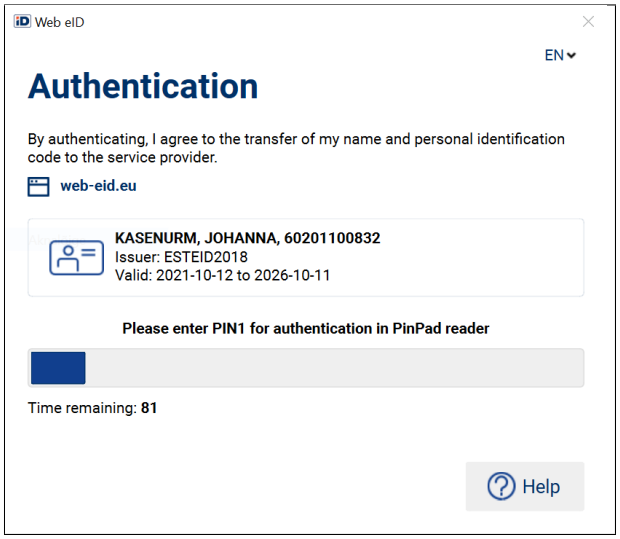


Figure 47. Web eID authentication window on Windows 10 when using a smart card reader with a pin pad

7.1.2 Darkening background of Web eID window

Darkening the background when the Web eID authentication window is shown, similar to what is implemented on Windows User Account Control, is another possible countermeasure (see Figure 48). This is a very noticeable change that can not be replicated in a picture-in-picture attack. A malicious website can only control the page content area and could darken the website behind the fake window, but it would not be able to darken the entire screen. Therefore, the user could easily detect spoofed Web eID windows if the entire background is not darkened. For such a measure to be effective, users would have to be trained to expect a darkened background and act suspiciously if a Web eID window appears without the background being darkened. The same applies to other measures discussed above.

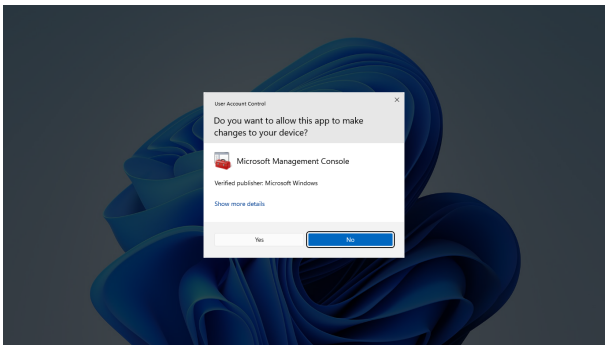


Figure 48. Windows User Account Control window on a darkened background [34]

7.1.3 Positioning the Web eID authentication crossing the line of death

Another possible countermeasure would be always positioning the Web eID authentication window crossing the line of death (see Figure 49). Since it is impossible to position a fake window outside the page content area, it could alert the user if the authentication window is positioned below the line of death. However, this difference is not drastic and requires attentiveness from the user. Furthermore, it may feel strange for the users to see the Web eID window positioned on the top edge of the screen rather than centred, especially with bigger screen resolutions.

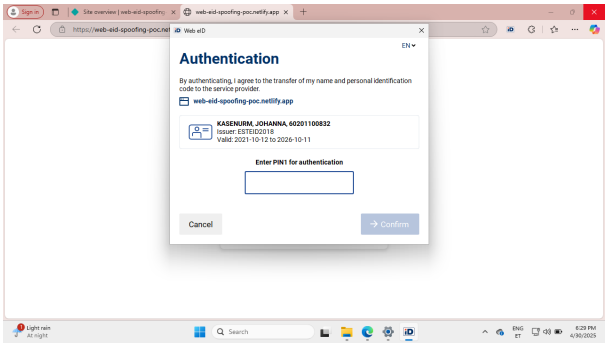


Figure 49. Web eID authentication window crossing the line of death on Windows 11

7.1.4 Minimising the browser window

Minimising the browser window when the Web eID authentication window is shown could also be an effective countermeasure (see Figure 50). Since it would be impossible to replicate this functionality in a picture-in-picture attack, the users would be able to detect the attack. However, this countermeasure is drastic and could be inconvenient for the users, as users would have to reopen the browser window after authentication. Also, users might prefer to see the website where they authenticate during the authentication process.

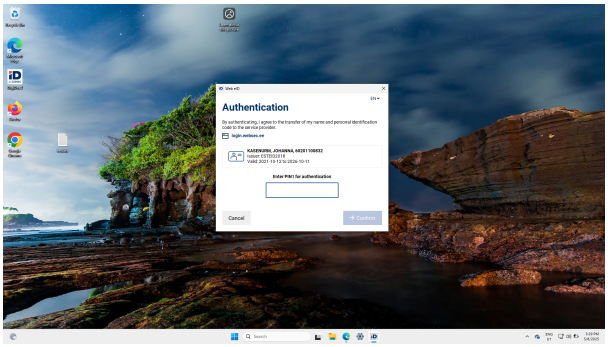


Figure 50. Web eID authentication window with minimised browser window on Windows 11

7.2 Raising awareness

Raising user awareness is another possible strategy for mitigating a picture-in-picture attack against the Web eID authentication window. A picture-in-picture attack has several limitations that would enable its detection. However, this requires knowledge and attentiveness from the users.

One thing the user can check to detect a picture-in-picture attack is moving the authentication window outside the page content area, over the line of death. Since a malicious website can't display content outside the page content area, moving a fake authentication window outside the

current website is impossible. However, the legitimate Web eID authentication window is an application window and can be dragged across the entire screen.

Additionally, when the legitimate Web eID authentication window is opened, a new process is started, resulting in an icon in the operating system's taskbar. In the case of a picture-in-picture attack, no new process is started, so there is also no icon in the taskbar. Therefore, users can use this difference to successfully identify a picture-in-picture attack. However, it should be noted that detecting the absence of a UI element is often more challenging than detecting the presence of an unexpected one.

Another difference between the legitimate Web eID authentication process and a picture-in-picture attack is that the browser window loses focus when the legitimate window opens. In the case of a picture-in-picture attack, the browser window is focused during the entire authentication process. This is visually marked by changing the colour of the browser's title bar. However, the colour change is only slight and is very difficult to notice.

Furthermore, customising the OS theme could also be one solution, as this makes replicating the Web eID authentication window more difficult. However, this still requires great attentiveness from the user. The user needs to be familiar with how the customisation affects application windows and be able to notice if anything is different. If the changes in the style of the Web eID authentication window are not drastic, the user could still fail to differentiate between legitimate and fake windows (as this was observed in the user study).

User-centric measures can be used to reliably detect a picture-in-picture attack against the Web eID authentication window, but they require knowledge and attentiveness from the user. In order to be able to detect a picture-in-picture attack, the user would need to understand the limitations of this attack and use them to differentiate between the legitimate and fake authentication windows. Spreading this knowledge throughout the general public could prove to be difficult. Furthermore, users tend to become less attentive when completing routine tasks and may still fail to detect a picture-in-picture attack. Therefore, relying only on user education to detect a picture-in-picture attack against the Web eID authentication window might not be a reliable solution, especially since the user study presented in this thesis shows that even the most educated and security-sensitive group of people might fall for the attack.

7.3 Conclusion

Several countermeasures can be applied to help prevent a picture-in-picture attack against the Web eID authentication window. The most effective countermeasure would be the use of smart card readers with PIN entry, but not all users possess this type of card reader. Furthermore, a user could still fail to detect the picture-in-picture attack if the malicious website displays a fake authentication window with a PIN entry field.

Technical countermeasures can be used to add visually noticeable functionality that is impossible to replicate in a picture-in-picture attack. These make the legitimate authentication and fake windows visually different and can help users to easier detect an attack. However, these measures can be inconvenient for the users. Picture-in-picture attacks also have several limitations that the user can use to test the legitimacy of the authentication window. However, this requires awareness and attentiveness from the user and is not a reliable solution.

8. Conclusion and future work

The thesis aimed to test the feasibility and effectiveness of a picture-in-picture attack against the Web eID authentication window. For this purpose, the legitimate authentication window was analysed in different operating systems and screen resolutions using different themes and scale factors. Based on the analysis, a fake window was created that enabled the capture of PIN1. The fake window was customised to almost identically replicate the legitimate authentication window on Windows 11, macOS Monterey, and Ubuntu 22.04 in both OS themes—dark and light.

The created fake authentication windows were used in a proof-of-concept web application demonstrating a picture-in-picture attack against the Web eID authentication window. The web application uses legitimate authentication with Web eID to obtain information from the authentication certificate stored on the user's ID card and simulates page loading. On the second authentication attempt, the web application displays one of the fake windows based on the user's operating system and theme.

A user study was conducted to test the effectiveness of the proof-of-concept implementation of a picture-in-picture attack against the Web eID authentication window. The study had ten participants with a strong IT background. The results showed that only two participants could detect the picture-in-picture attack. This proves that this type of attack can be difficult to detect even for users with technical knowledge.

Possible future developments of this work could include detecting the user's operating system version by implementing server-side code. This would enable the personalisation of the attack further by creating more accurate fake authentication windows.

Furthermore, several improvements could be implemented to make the attack more realistic. For example, the "Waiting for card" window could be replicated and displayed before the fake authentication window. The web application could also adjust the size of the fake window according to the page zoom and enable the users to change the language.

Additionally, the same techniques could be used to implement a picture-in-picture attack targeting the Web eID signing window and TLS CCA authentication window. It would also be possible to test different attack scenarios. For example, a malicious website could simulate an incorrect PIN entry and prompt the user to re-enter the PIN in a fake authentication window, or, after successful authentication, request a digital signature by displaying a fake Web eID signing window.

References

[1] RIA. RIA recommends the introduction of Web eID in e-services. https://www.id.ee/en/article/ria-recommends-the-introduction-of-web-eid-in-e-services/ (May 5, 2025).

[2] Jackson C., Simon D. R., Tan D. S., and Barth A. An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks. *Financial Cryptography and Data Security*. Ed. by Dietrich S. and Dhamija R. Vol. 4886. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 281–293. DOI: 10.1007/978-3-540-77366-5_27. http://link.springer.com/10.1007/978-3-540-77366-5_27 (May 5, 2025).

[3] mr.d0x. Browser In The Browser (BITB) Attack. https://mrd0x.com/browser-in-the-browser-phishing-attack/ (May 5, 2025).

[4] Das D. What Is a Browser-in-the-Browser Attack and How Can You Protect Yourself? June 2022. https://www.makeuseof.com/what-is-browser-in-the-browser-attack/ (May 5, 2025).

[5] Grustniy. Browser-in-the-browser attack: a new phishing technique. Apr. 2022. https://www.kaspersky.com/blog/browser-in-the-browser-attack/44163/ (May 5, 2025).

[6] Keski-Pukkila P. Watch: Sophisticated new credential harvesting attack uses fake popups and fake Microsoft Windows OS. Jan. 2022. https://hoxhunt.com/blog/credential-harvesting-fake-popups-microsoft-windows-os (May 5, 2025).

[7] Singh S. D. BITB (browser in the browser) Attack. Apr. 2022. https://infosecwriteups.com/bitb-browser-in-the-browser-attack-e2008c405701 (May 10, 2025).

[8] Web eID. web-eid/web-eid.js. Apr. 2025. https://github.com/web-eid/web-eid.js (May 5, 2025).

[9] Web eID. web-eid/web-eid-webextension. Feb. 2025. https://github.com/web-eid/web-eid-webextension (May 5, 2025).

[10] Web eID. web-eid/web-eid-app. Apr. 2025. https://github.com/web-eid/web-eid-app (May 5, 2025).

[11] Web eID. web-eid/web-eid-system-architecture-doc. Jan. 2025. https://github.com/web-eid/web-eid-system-architecture-doc (May 5, 2025).

[12] Web eID. web-eid/web-eid-authtoken-validation-java. Apr. 2025. https://github.com/web-eid/web-eid-authtoken-validation-java (May 5, 2025).

[13] Saurabh K. Top 10 Internet Browsers: Most Popular Web Browsers Reviewed. Feb. 2024. https://technumero.com/most-popular-web-browsers/ (May 5, 2025).

[14] StatCounter. Desktop Operating System Market Share Estonia. https://gs.statcounter.com/os-market-share/desktop/estonia (May 5, 2025).

[15] StatCounter. Desktop Windows Version Market Share Worldwide. https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide (May 5, 2025).

[16] Statistics log files for fr.archive.ubuntu.com / ubuntu.lafibre.info. https://fr.archive.ubuntu.com/stats/stats_of_day-457.html?version=last (May 5, 2025).

[17] Steiner T. Let there be darkness! □ Maybe… Aug. 2019. https://medium.com/dev-channel/let-there-be-darkness-maybe-9facd9c3023d (May 5, 2025).

[18] West J. What is my screen resolution - Check your Display Device. https://screenresolutiontest.com/screenresolution/?utm_source=chatgpt.com (May 5, 2025).

[19] StatCounter. Screen Resolution Stats Estonia. https://gs.statcounter.com/screen-resolution-stats/all/estonia (May 5, 2025).

[20] ericlaw. The Line of Death. Jan. 2017. https://textslashplain.com/2017/01/14/the-line-of-death/ (May 5, 2025).

[21] Suzanne M. Zoom, zoom, and zoom. July 2024. https://www.oddbird.net/2024/07/09/zoomies/ (May 5, 2025).

[22] Bright MLS. Quick Tip: Adjusting Your Browser's Zoom Level. https://support.brightmls.com/s/article/Quick-Tip-Adjusting-Your-Browsers-Zoom-Level (May 6, 2025).

[23] MDN Contributors. Navigator: platform property - Web APIs | MDN. Feb. 2025. https://developer.mozilla.org/en-US/docs/Web/API/Navigator/platform (May 5, 2025).

[24] MDN Contributors. User-Agent - HTTP | MDN. Mar. 2025. https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/User-Agent (May 5, 2025).

[25] WhatIsMyBrowser. Introduction to the Integration Guide. https://developers.whatismybrowser.com/api/docs/v3/integration-guide/introduction/ (May 5, 2025).

[26] Microsoft. Windows 10 supports ends on October 14, 2025 - Microsoft Support. https://support.microsoft.com/en-us/windows/windows-10-supports-ends-on-october-14-2025-2ca8b313-1946-43d3-b55c-2b95b107f281 (May 5, 2025).

[27] Web eID. web-eid-app/src/ui/images at main · web-eid/web-eid-app. en. https://github.com/web-eid/web-eid-app/tree/main/src/ui/images (May 5, 2025).

[28] RIA. Home. https://www.id.ee/en/ (May 5, 2025).

[29] Web eID. web-eid-app/src/ui/dialog.ui at main · web-eid/web-eid-app. en. https://github.com/web-eid/web-eid-app/blob/main/src/ui/dialog.ui (May 5, 2025).

[30] Web eID. web-eid-app/src/ui/fonts at main · web-eid/web-eid-app. en. https://github.com/web-eid/web-eid-app/tree/main/src/ui/fonts (May 5, 2025).

[31] Urushima K. kjur/jsrsasign. May 2025. https://github.com/kjur/jsrsasign (May 5, 2025).

[32] MDN Contributors. prefers-color-scheme - CSS: Cascading Style Sheets | MDN. Apr. 2025. https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-color-scheme (May 7, 2025).

[33] Paršovs A. Estonian Electronic Identity Card and its Security Challenges (2021), p. 45.

[34] Pamnani V., DarrenOMalleyMSFT, Matarazzo P., and Mauerer H. How User Account Control works. Apr. 2025. https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works (May 8, 2025).

License

I, **Johanna Kasenurm**,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis
   Web eID Authentication Window Spoofing Proof-of-Concept,
   supervised by **Arnis Paršovs, PhD**;
2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;
3. am aware of the fact that the author retains the rights specified in points 1 and 2;
4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Johanna Kasenurm

May 15, 2025