TECHNICAL UNIVERSITY OF CRETE

EXPLORING PHISHING-BASED THREATS IN THE USE OF SSO AUTHENTICATION ON THE WEB

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DIPLOMA OF ELECTRICAL AND COMPUTER ENGINEER

COMMITTEE

SUPERVISOR PROF. SOTIRIOS IOANNIDIS

PROF. APOSTOLLOS DOLLAS

PROF. NIKOLAOS GIATRAKOS

AUTHOR

PANAGIOTIS LEONTIS

School of Electrical & Computer Engineering

Microprocessor & Hardware Laboratory

CHANIA, SEPTEMBER 2024

This page intentionally left blank.

Acknowledgements

My supervisor, Prof. Sotirios Ioannidis, has my deepest gratitude for allowing me to work on this thesis project. I would also like to thank Panagiotis Ilia, Konstantinos Drakonakis and Michalis Diamantaris for their guidance and all their support and availability during the past months. Their understanding and contribution was substantial for the completion of this research. Special thanks to Prof. Apostollos Dollas and Prof. Nikolaos Giatrakos for evaluating my work and serving on the thesis committee. Finally,I want to sincerely thank my family and friends for their unwavering understanding, support, and encouragement during my academic career. Their unshakable faith in me has been a pillar of strength, encouraging me to keep going in the face of difficulty and helping me to realize my goals. Their presence, feedback and contribution have shaped me to this day.

Panagiotis Leontis
Chania 2024

This page intentionally left blank.

Abstract

This thesis delves into the exploration of Phishing threats in the use of Single Sign-On authentication on the web. Single sign-on (SSO) systems have become an integral part of modern digital authentication, providing users with easy access to multiple services with single credentials. However, we considered that phishing attacks targeting SSO login systems pose a serious risk to user security and undermine the effectiveness of these systems. Because they are always changing, hackers have mastered the art of deceiving unsuspecting people and avoiding detection. They create the impression that the user's personal information will be secure by impersonating a trusted provider. Therefore, there is a need for these threats to be more studied and recognizable. For this reason, we tried to act as an attacker and see how feasible it is to create such an attack by creating mock-up pages and interacting with them. Next step in our work was conducting a large scale experiment to evaluate if this kind of attack exists on the web. This goal was achieved by developing an automation tool to navigate through various domains and perform all the steps involved a login procedure using SSO as well as to discover any suspicious phishing attempt. After evaluating our approaches we wanted to be able to protect the user. Based on that a Chrome extension was implemented to be loaded to the user's browser and handle these cases. In this way, we could provide him with real-time defense if any malicious redirection is identified.

Keywords: Single Sign-On, Phishing, Detection

This page intentionally left blank.

Περίληψη

Στόχος αυτής της έρευνας είναι η ανίχνευση απειλών Phishing κατα τη χρήση του Single Sign-On στο διαδίκτυο. Τα συστήματα Single Sign-On (SSO) έχουν γίνει ένα αναπόσπαστο μέρος του σύγχρονου ψηφιακού ελέγχου ταυτότητας, παρέχοντας στους χρήστες εύκολη πρόσβαση σε πολλαπλές υπηρεσίες χωρίς την χρήση πολλών κωδικών. Θεωρήσαμε λοιπόν πως επιθέσεις για την κλοπή αυτών των στοιχείων στα συστήματα σύνδεσης SSO αποτελούν σοβαρό κίνδυνο για την ασφάλεια των χρηστών και υπονομεύεται η αποτελεσματικότητά και η χρησιμότητά τους. Δεδομένου ότι οι χάκερ εξελίσσονται συνεχώς, έχουν γίνει ειδικοί στο να εξαπατούν ανυποψίαστους χρήστες και να παραμένουν απαρατήρητοι. Υποδύοντας έναν έμπιστο πάροχο, π.χ το Facebook, εμπνέουν εμπιστοσύνη για την ασφάλεια των στοιχείων εισόδου που χρησιμοποιεί ο χρήστης. Επομένως, υπάρχει ανάγκη να μελετήσουμε περισσότερο αυτές τις απειλές και να μπορούμε να τις αναγνωρίζουμε. Αυτο μας οδήγησε στο να δράσουμε ως ένας κακόβουλος χρήστης και να αξιολογήσουμε κατα πόσο είναι εφικτή μια τέτοια επίθεση. Έτσι φτιάξαμε mock-up σελίδες ώστε να στήσουμε ένα τέτοιο περιβάλλον.Το επόμενο βήμα ήταν να αναζητήσουμε κατά πόσο υπάρχουν τέτοιες απειλές στο διαδίκτυο. Γι αυτό το λόγο αναπτύχθηκε ένα εργαλείο αυτοματισμού το οποίο εκτελεί όλα τα βήματα που περιλαμβάνει μια διαδικασία σύνδεσης με SSO καθώς και ανακαλύπτει κάποια ύποπτη προσπάθεια κλοπής των στοιχείων του χρήστη.
Αφού δοκιμάσαμε τις προσεγγίσεις μας, τρέχοντας εκτενή πειράματα θέλαμε να είμαστε σε θέση να προστατέψουμε το χρήστη. Βασισμένοι σε αυτό αναπτύξαμε ένα Chrome Extension το οποίο θα μπορεί να φορτωθεί στο πρόγραμμα περιήγησης του χρήστη και να χειριστεί αυτές τις ύποπτες υποθέσεις. Με αυτόν τον τρόπο, θα μπορούμε να του παρεχουμε άμυνα σε πραγματικό χρόνο αν εντοπιστεί κάποια κακόβουλη ανακατεύθυνση.

Keywords: Single Sign-On, Phishing, Ανίχνευση

Contents

This page intentionally left blank.

List of Figures

List of Listings

List of Tables

This page intentionally left blank.

1

Introduction

Cybercriminals constantly seek ways to steal confidential information, such as passwords and social security numbers, from internet users, making the current cyberspace environment full of persistent threats. They can obtain a user's login information as a result of this evil endeavor and utilize his account for a variety of malicious activities as described in [22, 31, 3, 20, 23]. As individuals navigate an ever-expanding digital ecosystem, characterized by a plethora of content and service providers (CSPs or SPs), the proliferation of user accounts becomes an unavoidable reality. Studies underscore this phenomenon, revealing that the average user contends with the management of different passwords, and the 65% of them reuses the same passwords as described in [17]. This burden not only engenders inconvenience but also engenders a pervasive phenomenon known as "password fatigue", as described in [27] wherein users experience mental exhaustion and frustration due to the complexities of password management. Consequently, the strain of juggling multiple passwords leads to cognitive overload, diminishing the overall user experience and impeding productivity. Furthermore, the increasing number of online accounts makes people more exposed to cyberattacks because having many passwords to remember raises the chance of security gaps and unsafe behavior. Given these obstacles, it is vital that creative solutions that improve cybersecurity defenses and expedite authentication procedures must be developed and put into place. These solutions are designed to make managing passwords easier, improve user experience, and strengthen digital defenses against ever-changing cyberattacks.

One significant step toward resolving the widespread issues with password management and authentication fatigue is the thoughtful deployment of Single Sign-On (SSO). By adopting SSO, users can enjoy seamless access to several services located across different domains after only needing to verify themselves once. There are three primary actors in the Single Sign-On model. The user who tries to access the system, the Service Provider (SP) or Relying Party (RP), and the Identity Provider (IdP). The IdP is responsible for the essential task of verifying user credentials. To safely validate users' identities, it employs advanced encryption techniques and robust authentication protocols. On the other hand, in order to relieve SPs of the laborious task of individual

user identification, the RP, sometimes referred to as the SP, depends on the IdP(s) for user identity verification. Because of this partnership, the authentication process is expedited, improving user satisfaction and lowering the possibility of security flaws that come with more conventional authentication techniques.
Moreover, SSO encourages scalability and interoperability in digital ecosystems, making it easier to integrate seamlessly with different platforms and service providers. Because of this compatibility, customers can access a wide range of services without having to go through laborious verification processes, which promotes productivity and efficiency. Furthermore, the widespread use of SSO has facilitated improvements in identity and access management (IAM), creating opportunities for more secure and user-centered solutions. Sensitive data is protected, user privacy is maintained, and the risk of unwanted access and data breaches is decreased with SSO's centralization of authentication procedures and enforcement of strict security measures. SSO's strategic deployment, in short, signals the beginning of a new era in digital authentication and completely transforms how users engage with online platforms and services [30].
However, we believe that Single Sign-On (SSO) will not be unnoticed by malicious actors. SSO systems could present an enticing opportunity for cybercriminals, offering a streamlined avenue to compromise user credentials in a singular breach, thereby granting unauthorized access to multiple platforms. This heightened attractiveness stems from the centralized nature of SSO, which consolidates authentication processes and simplifies access for users across various platforms.
In this research we introduce an attack that targets SSO systems. Our first goal was to evaluate if such an attack is feasible. Using a variety of strategies, malicious actors primarily rely on advanced social engineering techniques to obtain illegal access by extracting user credentials. The spread of phony emails and misleading websites stands out among these strategies as being especially pernicious since they are skilled at imitating reputable communications and interfaces and tricking users into inadvertently disclosing critical information. For our cause we used a Phishing toolkit, explained in section 3.1 to make mock-up pages and demonstrate how the user could fall victim on this scenario.
After the validation of this scenario we needed to conduct a large scale experiment and search for malicious websites targeting the login with SSO procedure. Eventually we have implemented an automation tool using JavaScript and Puppeteer, a framework offering automated interaction on the browser, presented in section 3.2. In pursuit of this objective, we leveraged a tool introduced in previous research [15], tailored to scrutinize vulnerabilities exclusively within implementations of legitimate Single Sign-On (SSO) providers, with a primary focus on Facebook. A functionality that we used, among others, was the automated identification of login pages, as it was a crucial step in our process. Afterwards, we enhanced this framework by adding more page interactions and making it more self-contained. We implemented devised mechanisms to interact with pop-up elements, as they frequently host login forms, necessitating visibility to facilitate detection. In addition, the number of Identity Providers that can be used for SSO have

increased to 9 in total. Moreover, we used Optical Character Recognition library to extract text content from images for the detection of non-conventional SSO buttons. Eventually we designed a parallel execution implementation in which individual browsers are instantiated for each identified SSO provider, ensuring meticulous scrutiny of SSO implementation and the better performance of our system. Subsequently, the login process is independently executed for each provider, with concurrent updates to log files containing screenshots, page sources, and details of identified providers for each visited domain. This comprehensive approach ensures meticulous scrutiny of SSO implementations across diverse platforms while enhancing the tool's efficacy in identifying and mitigating potential security threats The final goal of our tool it to visit various domains and detect any phishing attempt during the login with Single Sign-On. Since we were able to develop this kind of attack we wanted to provide a defense mechanism to protect the user. For this purpose we have implemented a Chrome Extension described in section 3.3 that can be loaded on the browser and detect any suspicious redirection and alert the user. In this way we provide a real-time safeguard to keep users away of accidentally sharing their login credentials.
A core limitation that we faced was researching an up-to-date and meticulously curated dataset of phishing websites to accurately reflect the prevailing SSO threat landscape. However, finding such a dataset presents inherent challenges, given the transient nature of these websites, which often vanish swiftly following reports. Moreover, considering the likelihood of divergent malicious implementations deviating from conventional patterns, our program necessitated robust capabilities to detect such anomalies.
The manual verification of each website entails a laborious and time-intensive process demanding specialized expertise in the field. Through rigorous research and iterative experimentation, this study has successfully devised a potent technique aimed at augmenting the efficacy of identifying and thwarting phishing attempts within the Single Sign-On framework. Consequently, users benefit from fortified defenses against malicious incursions. Furthermore, the integration of a dynamic runtime defense mechanism empowers the extension to proactively counter evolving phishing tactics in real-time, thereby furnishing users with preemptive measures to safeguard their sensitive information and mitigate potential risks effectively.

1.1 Motivation

The motivation articulated underscores the critical necessity for a robust mechanism adept at detecting and thwarting phishing attempts during Single Sign-On (SSO) login procedures. This imperative has propelled us to embark on a mission aimed at discerning patterns that indicate a malicious intent on the internet. Our objective has been to develop a comprehensive framework endowed with the capability to conduct large-scale experiments autonomously and provide a defense solution to the users.
Our motivation is deeply rooted in addressing a problem which could greatly affect individuals on a daily basis, exacerbated by the widespread adoption of SSO login

mechanisms. The convenience and efficiency offered by SSO systems have led to their widespread integration across numerous platforms, resulting in heightened exposure to potential security threats. Recognizing the impact of a potential threat, we are driven by a commitment to providing tangible solutions that enhance cybersecurity and protect user credentials from exploitation. Furthermore, our pursuit is fueled by a profound understanding of the escalating threats posed by cybercriminals, who perpetrate nefarious activities. Typosquatting represents a single example among various tactics used by malicious entities online. It involves registering domain names that closely mimic popular websites or brands but include slight typographical errors. This technique is relied on mistakes made by internet users, such as mistyped URLs or search queries, redirecting unsuspecting users to fraudulent websites. These deceptive sites often replicate the appearance and functionality of legitimate platforms to trick visitors into revealing sensitive information or unknowingly downloading malicious software. Exploiting common typing errors, typosquatters aim to generate revenue from redirected traffic or execute cyberattacks, posing significant threats to cybersecurity and user privacy.
Essentially, what drives us is a dedication to improving cybersecurity procedures and reducing the threats that users might face during the use of Single Sign-On. The final goal of our research is to make a significant contribution to the ongoing battle against cyberthreats and, in the end, achieve a more safe and secure online environment.

1.2 Contributions

In summary, we make the following research contributions:

- Introduced an attack vector that targets SSO systems. By using a phishing toolkit we managed to demonstrate this type of attack as a real world scenario. Our methodology included creating mock-up pages and interacting with them as a normal user would do.
- Evaluated the existence of Single Sign-On Phishing Attacks by conducting a large scale experiment. For this cause an automation tool was built, offering website interaction and malicious attempts detection. The tool was implemented by enhancing a predefined automation framework called SAAT to meet our requirements. This framework was initially designed to scrutinize vulnerabilities exclusively within implementations of Facebook Single Sign-On (SSO).
- Offering a real-time defense mechanism to the users while using SSO login. By developing a functionality that has been seamlessly integrated into a Chrome Extension, we provide with runtime protection by promptly identifying and flagging any suspicious activity. This tool can be loaded into user's browser and alert him when a malicious redirection is recognized during the Single Sign-On login.

1.3 Thesis Outline

This chapter introduced the problem of phishing, specifically when targeting Single Sign-On implementations. Additionally, it set the foundation for the drive behind the research presented in this study and encapsulated its primary contributions. The subsequent sections of this paper are structured as follows:
In Chapter 2 we elucidate the theoretical underpinnings essential for comprehending the scope of this research. In summary, we delve into 3 main topics, which are Phishing, Single Sign-On and Chrome Extensions. In Chapter 3, we expound upon the methodology employed and provide intricate technical insights into our work. Subsequently, we meticulously register and analyze the outcomes of our experiments in Chapter 4. Moreover, Chapter 5 delves into a review of related works that have contributed to the field of security, comparing side by side and contrasting our research endeavors within the broader context. Finally, we encapsulate our research findings, including an appraisal of our limitations, and delineate prospective avenues for future exploration aimed at achieving enhanced outcomes in Chapter 6.

2

Background

Our goal in this section is to provide a detailed explanation of the various stages involved in a Phishing Attack, Single Sign-On (SSO) techniques, and Google Chrome Extensions integration. Phishing is the term for an attempt to obtain private information in order to use it or sell it.This information is usually in the form of bank account information, usernames, passwords, or other critical data. An attacker tricks the victim by posing as a reliable source and making an alluring request. It is critical to comprehend how phishing attacks work since they take advantage of human weaknesses to obtain financial and personal information without authorization. In the meantime, by allowing access to numerous services using a single set of credentials, Single Sign-On (SSO) systems have simplified user authentication procedures. But SSO implementations are not impervious to security flaws, therefore strong protections against phishing efforts aimed at SSO-enabled platforms are required. Furthermore, the integration of Google Chrome extensions introduces a pivotal dimension in enhancing browser functionalities and security measures. These extensions, which make use of Chrome's extensibility, are essential for improving the user experience and strengthening security against online dangers. By analyzing these interrelated domains, this research aims to clarify the complex interactions among phishing assaults, SSO procedures, and Google Chrome extensions, promoting a better understanding of modern cybersecurity paradigms.

2.1 Phishing

Phishing is a common type of cyberattack that targets people via phone calls, sms, emails and other analogous actions. The goal of this kind of attack is to deceive the target into revealing his sensitive information such us financial information or system login credentials.
As a common type of social engineering phishing includes psychological manipulation and deception whereby threat actors mislead users into performing specific actions. These behaviors frequently entail, downloading and installing dangerous software, and following links to fake sites.
In order to successfully defend against phishing, we must fully understand the main

steps as well as the characteristics of phishing attacks.
Figure 2.1 depicts the steps of a typical phishing scenario. First, an attacker uses a website and by mimicking its appearance tries to make impossible for the average user to tell the difference between the original and his phishing website. The attacker then uses social engineering to persuade the user to take action by sending communications (e.g. spam emails) and convincing them to click on a link that takes them to a phishing website. After being tricked successfully, the victim opens the website and enters sensitive data, like credit card numbers or account credentials. It is a common practice to display a comforting confirmation message to victims in order to reduce post-attack suspicions. Finally the victims information is sent back by the phishing site to the attacker who will try to use it for a variety of reasons.

Phishing Techniques

There are three most used techniques to achieve information elicitation.

- **Malicious Web Links:** Phishing links direct users to fake websites or websites that contain malware, or malicious software. Malicious links can be included in logos and other pictures in an email and can appear to be trustworthy.
- **Malicious Attachments:** They appear as legitimate file attachments but they are actually infected with malware that can constitute a threat.
- **Fraudulent Data Entry Forms:** These methods make use of fictitious forms that ask consumers to enter private data, including phone numbers, credit card numbers, and user IDs. Once users provide the information, fraudsters may exploit it for identity theft and other illicit actions.

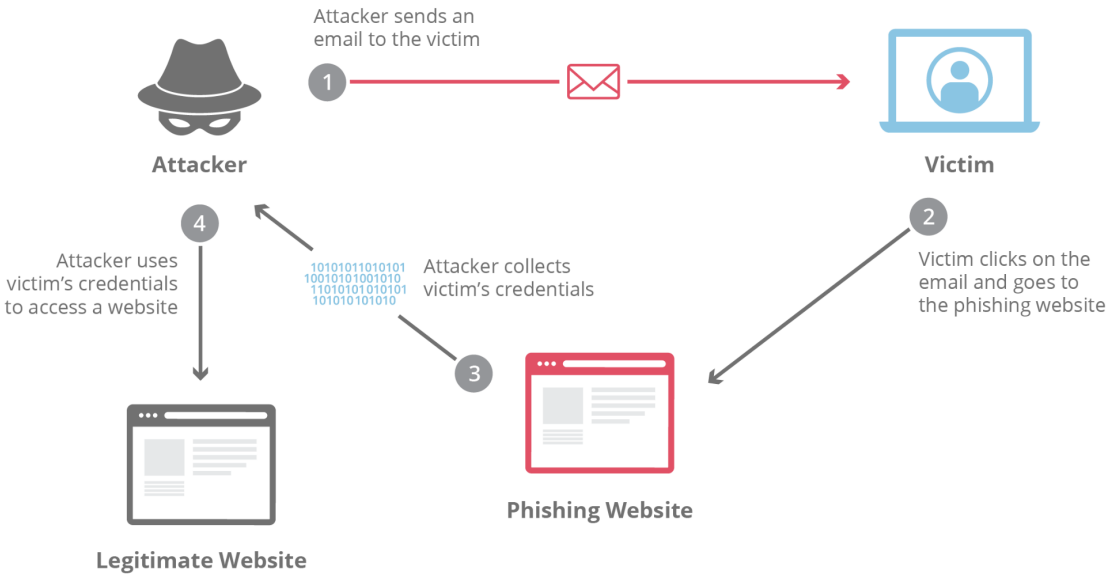


Figure 2.1: *This diagram illustrates a standard flow for phishing attacks. [10]*

Most Targeted Industries

Since financial gain is the major motivation behind most phishing attacks, attackers typically target particular companies that either hold credit card data or have the resources to make significant payments. Either the entire organization or certain users could be the target.

As depicted in Figure 2.2 social media and financial institutions are the two main targets of these malicious users as of second quarter of 2023.

Most Impersonated Brands

Attackers employ well-known brands copies in an attempt to fool as many people as possible. Recipients gain trust from well-known brands, which increases the likelihood of a successful phishing. According to a new study by Abnormal Security, presented by [8], which examined credential phishing and brand impersonation trends in the first half of 2023, Microsoft was the brand most frequently used as camouflage in phishing exploits, as shown in Table 2.1. However, any well-known brand can be used in phishing.

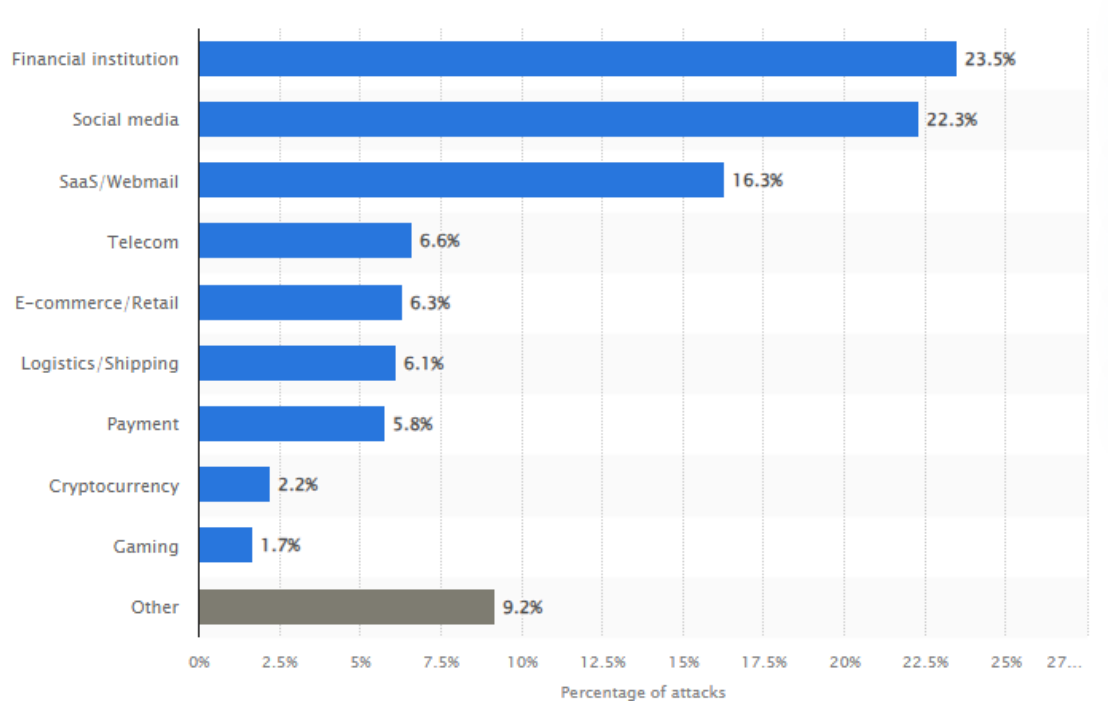


Figure 2.2: *Illustration of Most Targeted Industries by Phishing Attacks. (Q2 2023) [29]*

Table 2.1: *Most Impersonated Brands by Phishing Attacks. (Abnormal Security 2023) [8]*

Brand	% of Total Attacks	Industry
Microsoft	4.31%	Technology
PayPal	1.05%	Finance
Facebook	0.68%	Social Media

Table 2.1 continued from previous page

Brand	% of Total Attacks	Industry
DocuSign	0.48%	Technology
Intuit	0.39%	Finance
DHL	0.34%	Shipping
McAfee	0.32%	Technology
Google	0.30%	Technology
Amazon	0.27%	Retail
Oracle	0.21%	Technology

2.2 Single Sign-On

Single sign-on (SSO) is a crucial authentication method in modern digital settings that enhances user experience, bolsters security, and simplifies access. SSO eliminates the need for users to continuously input login credentials by allowing a single authentication to be used for multiple networked systems, apps, or services. This feature enhances user experience by being convenient and easing the cognitive load of managing multiple credentials.
SSO is founded on the trust established between an Identity Provider (IdP) and a Service Provider (SP) application. This connection is created through the exchange of a certificate between these two providers. This certificate can be used to sign identity information that is being sent from the IdP to SP so that the second knows it is coming from a trusted source. This data takes the form of tokens which contain identifying bits of information about the user's username or email address.
The login flow should resemble the following pattern shown in Figure 2.3:

1. Users wants to use a service and they must verify their identity.
2. As part of a request for user authentication, the Service Provider delivers a token to the SSO system (IdP) that contains certain user information.
3. A procedure verifies if the user is already been authorized in order to access be granted to him by the Service Provider. In such case we move to step 5.
4. If he hasn't previously, the user will be asked to log in. There are methods to utilize this, such as passwords, biometrics, and multi-factor authentication (MFA).
5. A verification token will be returned from the Identity Provider to the Service Provider following a successful authentication.
6. This token passes through the user's browser to the Service Provider.
7. This confirmation is validated according to the trust relationship that was set up between the two Providers during the initial configuration.
8. Access is granted to the User.

In a case where user tries to access a different website, it would have to possess a similar trust relationship configured with the SSO solution and the authentication flow

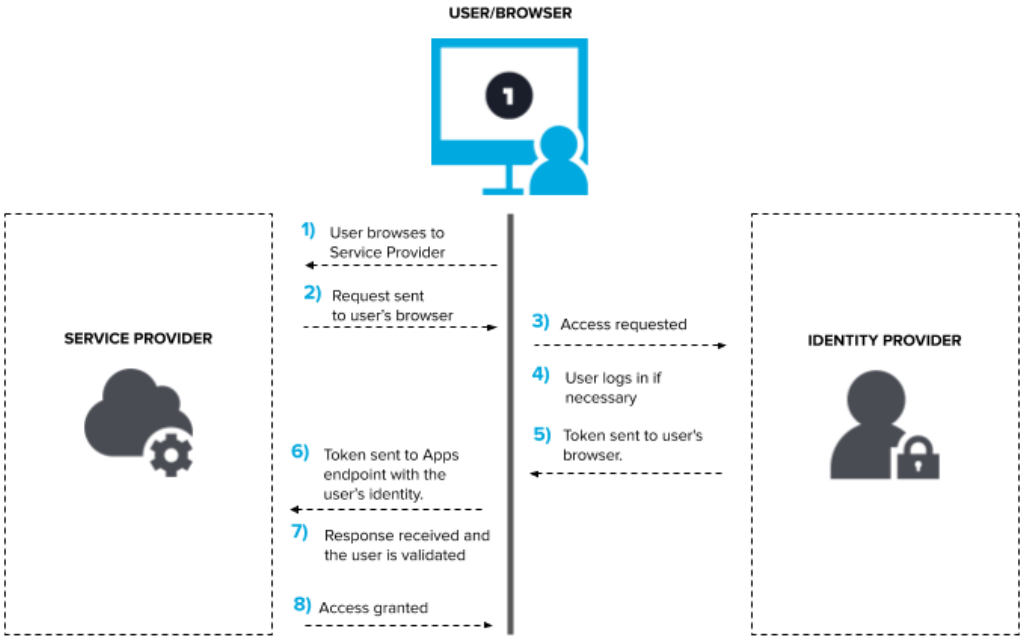


Figure 2.3: *Illustration of user authentication via an Identity Provider. [24]*

would follow the same steps.
There are two basic protocols that are mainly applied in SSO implementations:

- **Security Assertion Markup Language (SAML)**: SAML is an XML-based totally open popular for exchanging authentication and authorization information among parties, typically among an identification provider (IdP) and a provider company (SP). In the SAML-based SSO workflow, when a user attempts to get right of entry to a carrier, the provider company redirects the person to the identity issuer for authentication. Upon a hit authentication, the identification company generates a SAML announcement containing the user's identification information and sends it lower back to the provider provider, granting get right of entry to to the requested useful resource.
- **OpenID Connect:** OIDC, short for OpenID Connect, represents a straightforward identity layer that complements OAuth 2.0 protocol, designed specifically for authentication. It provides a standardized way for clients to verify the identification of end-users based totally on the authentication carried out by an authorization server. In OIDC-based SSO, the authentication flow involves the consumer (or relying party), the OpenID provider (IdP), and the cease-consumer. The consumer initiates the authentication request, which is then redirected to the OpenID issuer. After a hit authentication, the OpenID company returns an ID token. As a final step, the client application can use this token, which contains vital information about the user and the authentication event, to confirm the user's identity and move forward with access control decisions.

Below we present two different payloads: the first one contains the structure of a Security Assertion Markup Language (SAML) token (Listing 2.1), and the second one

shows an Identity Token (ID Token) (Listing 2.2). These payloads serve as illustrative paradigms, elucidating the underlying data structures and content associated with these authentication tokens.

SAML Announcement Structure

- **<saml:Assertion>:** Root Element of the SAML assertion
- **<saml:Issuer>:** Identifies the issuer of the assertion (IdP)
- **<saml:Subject>:** Contains information about the user
  - **<saml:NameID>:** Specifies the identifier for the subject (e.g. email address)
- **<saml: Conditions>:** Defines the conditions under which the assertion is valid.
  - **<saml:AudienceRestriction>:** Identifies the assertion's intended audience (the service provider).

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="myID"
↪  IssueInstant="2024-02-20T12:00:00Z" Version="2.0">
  <saml:Issuer>https://idp.example.com</saml:Issuer>
  <saml:Subject>
    <saml:NameID>pleontis@example.com</saml:NameID>
  </saml:Subject>
  <saml:Conditions NotBefore="2024-02-20T12:00:00Z"NotOnOrAfter="2024-02-20T12:05:00Z">
    <saml:AudienceRestriction>
        <saml:Audience>https://acme.example.com</saml:Audience>
    </saml:AudienceRestriction>
  </saml:Conditions>
</saml:Assertion>
```

Listing 2.1: *SAML Announcement Example*

ID Token Structure

- **iss:** The issuer of the token (IdP)

```
{
    "iss": "https://idp.example.com",
    "sub": "1234567890",
    "aud": "https://acme.example.com",
    "exp": 1516239022,
    "iat": 1516238122,
    "name": "Panagiotis Leontis",
    "email": "pleontis@example.com",
    "nonce": "n-0S6_WzA2Mj",
    "auth_time": 1516238121
}
```

Listing 2.2: *ID Token Example*

- **sub:** The subject identifier, a unique identifier for the authenticated user
- **aud:** The audience, which specifies the intended recipient of the token, typically the client application
- **exp:** The expiration time of the token in UNIX timestamp format
- **iat:** : The issuance time of the token in UNIX timestamp format
- **name:** The name of the user
- **email:** Email address of the user
- **nonce:** A nonce value used to mitigate replay attacks
- **auth_time:** The user's authentication time, also expressed as a UNIX timestamp

We can infer from the description above that there is almost no difference in the procedures to be followed when logging in with SSO. Because of this, a malevolent user can identify and mimic these patterns.

2.3 Google Chrome Extensions

Chrome extensions are useful instruments that enhance the Google Chrome browser's capabilities and let users add new features and personalize their browsing. The manifest.json file, background scripts, content scripts, popup pages, and interactions with Chrome pages make up the fundamental parts of the Chrome extension. Below there is a basic illustration of the structure of a Chrome Extension in Figure 2.4

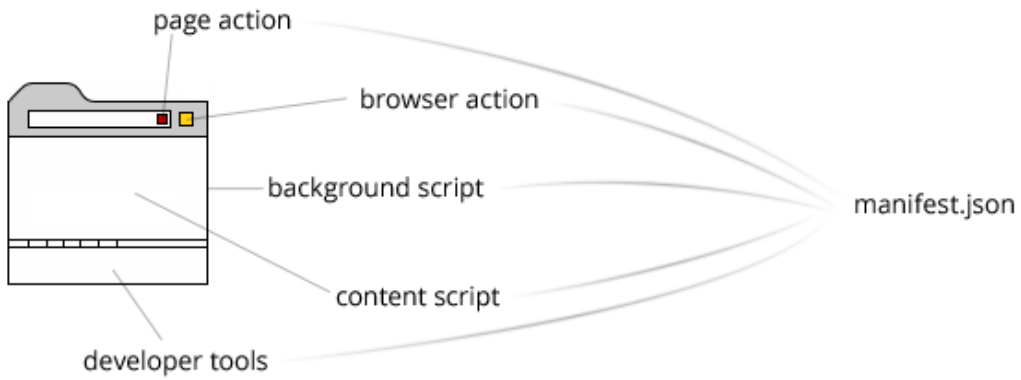


Figure 2.4: *This diagram illustrates the structure of a Chrome Extension. [11]*

- **Manifest.json:** This JSON-formatted file containing metadata. It serves as a comprehensive guide for understanding the extension's qualities and authorizations. This file mainly includes essential details like the extension's title, update, brief synopsis, graphics, and most significantly, its capabilities. Additionally, it outlines the necessary permissions for accessing distinct browser elements or APIs.
- **Content Scripts:** Content scripts are JavaScript files injected into web pages by the browser. Through their usage the extension is enabled to to interact with and

modify the content of those pages. The Document Object Model (DOM) of a web page provides content scripts with the ability to change elements, listen for events, and interact with background scripts.

- **Background Scripts:** JavaScript files known as background scripts operate independently of the browser, even when the browser window is minimized or the extension popup is closed. They perform tasks that require regular or periodic execution, such as text a managing events, executing network requests, or managing data storage. Background scripts have access to the entire Chrome API, allowing you to interact with browser elements such as tabs, windows, bookmarks, and more.
- **Popup Window:** Chrome extensions come equipped with elements that allow for user interface functionality. This may include features like popup windows or options pages, which enhance user interaction and customization. For example, by configuring the `popup.html` file, the extension's popup window can be designed with specific content and arrangement. Once the user clicks on the icon in the browser toolbar, the popup window will appear and can provide information, receive input from the user, and even initiate actions within the extension, offering an ease of use to him.

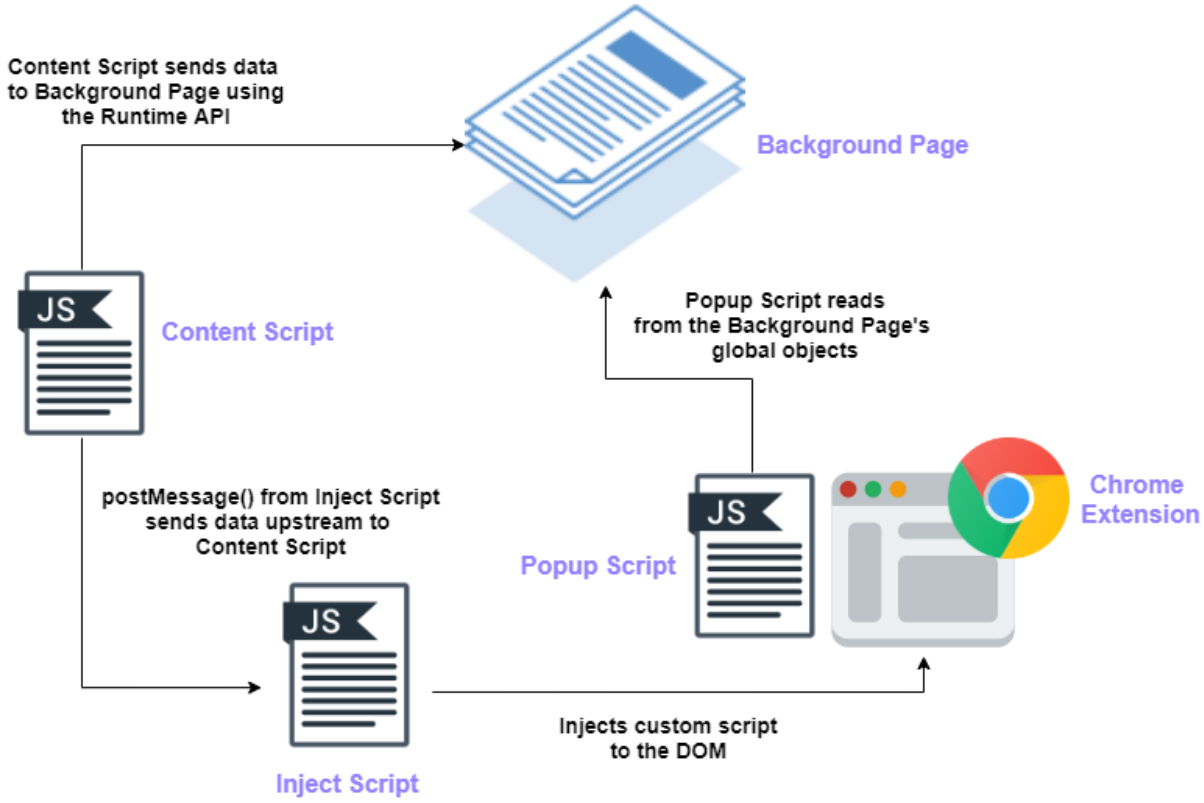


Figure 2.5: *This diagram illustrates the life cycle of a Chrome Extension. [12]*

As a user engages with a Chrome extension, a cohesive collaboration of various components drives the desired functionality as illustrated in Figure 2.5. Consider, for

instance, the simple action of clicking on the extension's icon. This activates the popup window, which is defined by popup.html and may display information pulled by the background script or allow the user to input data. Subsequently, the popup window and the background script communicate to carry out actions or retrieve additional information. But that's not all - the content script also plays an integral role throughout this process. It remains active, interacting with and making changes to the web pages visited by the user. By keeping an ear out for page events, manipulating DOM elements, and effectively communicating with the background script, the content script ensures a smooth and seamless experience overall.
Understanding how these components interact is key to develop and use Chrome extensions in an effective way.

3

Methodology

In the forthcoming section, we offer a comprehensive insight into the technical underpinnings of our research endeavors. First we analyze the steps on how we made mock-up Phishing pages in order to interact with them in an environment where the attack takes place. We delineate the step-by-step process involved in constructing an automation tool tailored to detect Single Sign-On (SSO) phishing attempts. This includes the integration of the existing SAAT framework, alongside extensions and modifications to enhance its functionality and efficacy. Additionally, we detail the development of a Chrome extension aimed at providing users with a proactive defense mechanism against phishing threats encountered during SSO login processes. Throughout this section, we provide a thorough exposition of the implementations utilized to bolster the robustness and accuracy of our solutions.

3.1 Implementing the Phishing Attack

Our goal was to create an SSO Phishing attack to evaluate how feasible it is in the real world. For this purpose, we employed a sophisticated toolkit known as SocialPhish [28], specifically designed to craft counterfeit websites emulating reputable brands. Further details regarding the usage of Socialphish can be found in Appendix B. Utilizing this tool, we meticulously constructed deceptive login pages, each representing various identity providers pivotal to our experiment.

As a first step we had to implement a fake login portal using `HTML` and `CSS` which seems like a legitimate page. Next step was the design of fake SSO button elements, using the same technologies, mimicking the original ones. We also proceeded on creating images illustrating a button to handle an unorthodox way of implementing a button. These kind of elements most of the time stay undetected due to their nature, as they require an Optical Character Recognition mechanism to be identified. Our methodology included 9 different SSO Providers and each fake element is illustrated in Figure 3.1. The final step for the implementation of the attack was adding these buttons on the login portal. On Figure 3.2 there is illustrated the fake login portal containing one of the fake SSO buttons as well ass the results after user's interaction.

After generating the fake SSO login page using the toolkit and embedding the URL in the fake SSO element, we had completed each step of the attack. Subsequently, we orchestrated a live simulation for each Identity Provider, assuming the role of an unsuspecting user inadvertently directed to these fraudulent pages during the authentication process. As the user tries to connect to a trusted provider, he does not understand that he is on a Phishing Website and the attacker through the toolkit can capture his credentials.

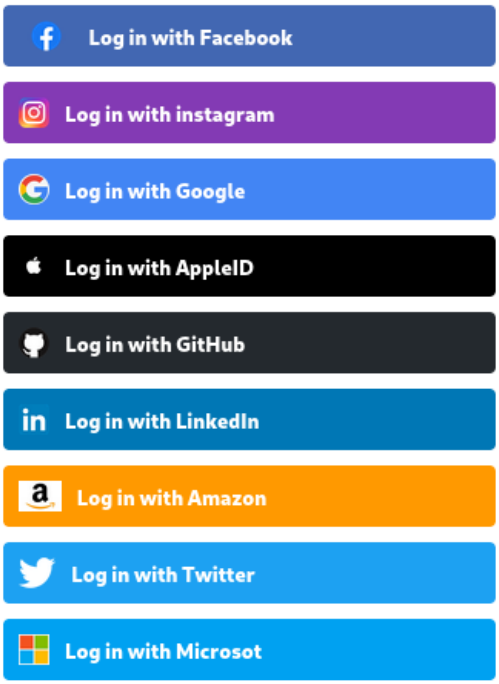
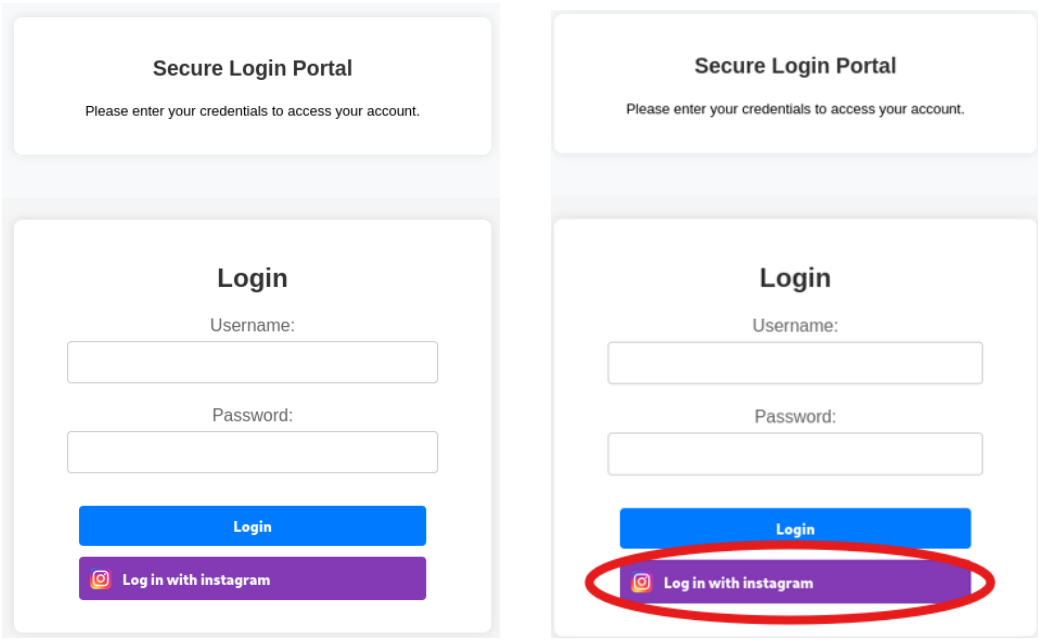


Figure 3.1: *Illustration of the fake SSO login elements implemented to conduct a real-time experiment and validate our defense mechanism.*

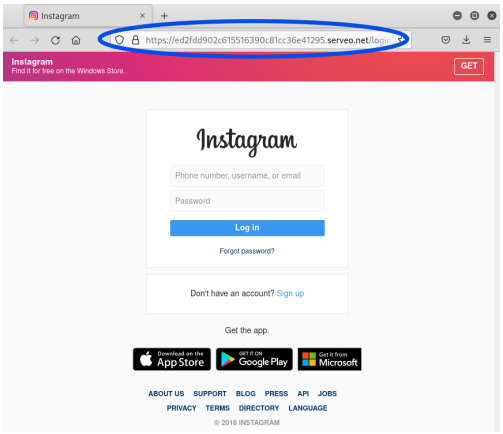
By following the above steps we managed to create an attack that targets SSO login and allowed us to see how a malicious user could implement it. We have noticed that by using commonly available tools and some additional sophistication, a malicious environment can be built which can fool a user into revealing his credentials.

3.2 Automation Framework

Our goal was to conduct a large scale experiment and evaluate the existence of threats targeting the SSO as described on the previous section. To achieve this goal we needed to develop an automation framework that will be able to visit different domains and to detect malicious patterns during the SSO login. In this section we analyze the functionality of the existing SAAT framework that we used for our needs in subsection 3.2.1 and the modifications we made in order to extend its capabilities in subsection 3.2.2.



(a) *User gets redirected to Phishing Website.* **(b)** *User selects log in with SSO.*



(c) *User gets redirected to fake SSO login page.*

Figure 3.2: *Illustration of the steps followed by the user during an SSO phishing attack*

3.2.1 Functionality Used from SAAT Framework

The tool was built upon Puppeteer [16] for orchestrating the automated interaction on the browser. This research was focused on building an automated framework that assesses whether relying parties that use Facebook as the IdP comply with secure practices and guidelines. [15]
In this section, we analyze the points and the functionality they include that seemed useful to us. Using these pre-implemented components enabled us to develop our framework for running a large-scale experiment.

Locating Registration Form

First we incorporated the functionality, which involved a crawler tasked with visiting registration pages and identifying the sign-up section. The approach employed involved scrutinizing all `<form>` elements and employing keyword matching to locate sign-up forms. If the crawler encountered a form element without matching keywords, it selected it only under specific conditions: when it was the sole form on the page and lacked search or login-related keywords.
In cases where the crawler failed to detect forms, it searched for links potentially leading to account creation pages, a useful tactic for websites with indistinguishable login and registration URLs, as well as single-page applications featuring sign-up forms upon interaction. Upon locating forms, the crawler populated all visible non-hidden <input> fields with random information and checked all checkboxes. This initial process facilitated the detection of dynamically generated input elements based on other filled inputs, such as password confirmation fields. Subsequently, the crawler recorded visible inputs, determined the type of personal information required, and filled corresponding input fields accordingly. Similar strategies were applied to `<select>` and `<input type="radio">` fields, with random selection utilized if the type of information needed couldn't be identified. For each input type, a set of possible values was predefined in case some were rejected by the website.

Single Sign-On Workflow

The functionality utilized for SSO detection was implemented by leveraging the browser's Web Accessibility API within the tool. This API is designed to expose an interface for assistive technologies, providing a semantic representation of the user interface to convey crucial information across various platforms, particularly benefiting users with impairments. Additionally, it finds application in automated testing and UI automation, as seen in applications like password managers [9]. There is provided a code sample of HTML page tags and its corresponding Accessibility Tree in Appendix A.
The built framework uses Chromium's Accessibility API, which returns a web page's representation as a tree of objects and traverses the tree to look for nodes that contain SSO indications. Puppeteer's accessibility tree did not directly expose DOM nodes, they modified the corresponding `Accessibility` class to expose each node's unique

identifier, which is used for resolving the node that contains SSO-related content. Upon detecting SSO support, the tool proceeds with initiating the login process for the relying party (RP). This step includes interacting with different buttons on the page and handling different kinds of window redirections. The appropriate logic is implemented for different log-in workflows that might be encountered. These may include a popup window containing the IdP login page, a spawned window, or even a redirection in the current window. All these different scenarios required a sequence of steps ensuring a seamless interaction with the browser and completion of the log-in mechanism.

3.2.2 Enhancements Made on SAAT

After reviewing each section of the provided framework we focused on specific components that could be useful to our goals. Some of them had to be adjusted to meet the requirements of this research.

Login Page Detection and Popup Interaction

The initial tool was designed to operate on the login page as a starting point. This sets an obstacle to operating on a more autonomous way as it requires only specific URLs for each website to be used. Our goal was to improve this functionality, which would allow us to use the homepage of each website as a starting point for the automation framework. We managed it by adding a page interaction that would enable us to navigate at the login page or even make it visible in some cases. Many sites contain a login form inside a popup window that will be visible to the user after some specific actions on the website. The preceding tool could not identify such instances, as a popup window does not constitute a new URL that our automation can navigate to.
The implementation contains a recursive traverse through the DOM, examining each element. While using the already provided resources for this algorithm, we targeted clickable elements that contain a text that indicates a login step shown in Listing 3.1 In this way, we follow the framework's predefined logic and if no SSO indicators are found we search for the login page. If there is an outcome we repeat the process explained in the previous section.

```
1 login_regex = /(log(ged){0,1}|sign(ed){0,1})\-{0,1}\s{0,1}(in|on)/i
```

Listing 3.1: *Regular Expression tailored to login page*

Enrichment of the SSO providers dataset

As above mentioned the provided tool was implemented to detect flaws only on Facebook's SSO implementation. Our goal was to expand the scope and the usability of this framework, to match this experiment's needs.
The SSO workflow that the automation would follow was going to be applied to all different SSO Providers. After locating the Sign In form the tool searches for SSO

elements based on regular expressions and using the inner text and attributes (e.g `id`) of each element. Before our modification, the tool was capable of only detecting Facebook's SSO Login Button. Utilizing the established framework of implementation, enhancements were made to incorporate regular expressions for an additional eight prominent identity providers, encompassing Google, Apple ID, Twitter, and Microsoft, among others shown in Listing 3.3.

Execution Parallelism

To enhance the scalability of our framework in extensive experimentation, we introduced a parallel execution mechanism. Following the completion of preliminary steps to identify SSO elements, distinct instances of web browsers are spawned, each dedicated to a specific element. This approach enables concurrent logging procedures for individual Identity Providers, thereby eliminating the need to await the completion of others.

Logging Results and Website's Resources

Within the login procedure, we have instituted a comprehensive logging mechanism to meticulously track execution and outcomes. Initially, a directory is established for every accessed website, as shown in Figure 3.3, to store each capture. In a more specific vein, the structuring of our logging metrics follows this particular approach:

Page Directory

Within this designated directory, we meticulously preserve all the pertinent metrics that encapsulate the essence of a website, facilitating subsequent review and monitoring of our framework's performance. Initially, we initiate the process by capturing a screenshot of the website through the native functionality embedded within Puppeteer. This strategic approach enables us to conduct a thorough examination later, assessing the detection rate of Single Sign-On (SSO) elements and identifying any potential oversights. Additionally, we meticulously download the HTML code of the website, ensuring comprehensive coverage of its underlying structure. Within the resources directory, we meticulously catalogue each injected script found on the website, leveraging `Axios` to efficiently retrieve data from the script URLs. `Axios`, a popular JavaScript library, facilitates seamless data fetching from external resources, augmenting our data collection capabilities. Subsequently, we meticulously document all discovered SSO elements in the logs.txt file, accompanied by their respective mapped SSO providers and the URLs to which the SSO elements redirect users upon activation. This meticulous logging process ensures a comprehensive record of all relevant information for subsequent analysis and evaluation.

IDPs Directory

As elucidated earlier, the framework adheres to a parallel execution model for every Single Sign-On (SSO) element identified during each visit to a website. Each spawned execution establishes a dedicated directory corresponding to the mapped Identity Provider of the respective SSO element and autonomously organizes the logging metrics. These metrics, essential for evaluation and analysis, are systematically captured following the methodology outlined in the preceding subsection. In the event of encountering an error during execution, the domain of the website is recorded in a text file along with a description of the encountered error. Similarly, upon detection of any malicious activity, the domain is logged in a separate text file containing all identified malicious domains encountered during experimental executions. This meticulous logging process ensures comprehensive documentation of errors and malicious activities, facilitating subsequent analysis and mitigation efforts.

Through this methodical approach, we conducted a thorough evaluation of our framework's performance, scrutinizing its capability to detect Single Sign-On (SSO) elements within web pages and the accessibility of relevant data for subsequent analysis. It is important to highlight that phishing websites often have a transient presence due to their rapid identification and reporting of their threat status. Therefore, these vital metrics hold significant significance post-execution, ensuring the ongoing effectiveness and adaptability of our framework in addressing emerging threats.

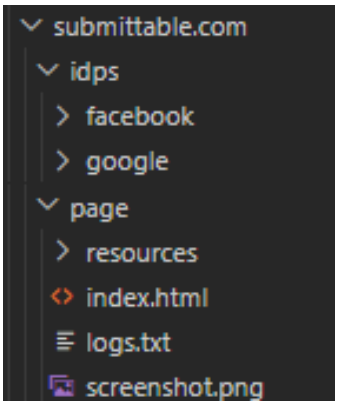


Figure 3.3: *Illustration of the structure of logging metrics that are captured for each visited website during the automated login procedure.*

3.3 Chrome Extension

This section presents a Chrome extension that was designed to offer runtime protection against phishing attempts that target SSO implementations. The extension comprises multiple components designed to proactively identify and mitigate potential threats encountered during browsing sessions.

3.3.1 Manifest.json: Extension Configuration

This file serves as the configuration file for our Chrome extension, defining its functionality, permissions, and resources. This section provides an overview of the key components and capabilities being specified within the manifest file.

1. **Version and Metadata**
    - **Manifest Version:** The extension utilizes Manifest Version 3, the latest version of the Chrome extension manifest format. It offers improved security, performance and reliability.
    - **SName and Version:** The extension is named "SSO Detection and Verification" with version "1.0", providing users with clear identification and versioning information.
2. **Persmission**
    - **Active Tab:** The extension is granted permission to interact with the currently active tab, enabling dynamic content analysis and user engagement.
    - **Web Navigation:** Permission is granted to monitor and analyze web navigation events, facilitating URL redirection validation and phishing attempt detection.
    - **Tabs:** Gives authorization to access information about browser tabs, supporting interaction and coordination between extension components.
3. **Background Script**
    - **Service Worker:** The background script is specified as "background.js", serving as the central component for coordinating actions, validating URL redirections, and responding to messages from content scripts.
4. **Content Scripts:**
    - **Matches All URLs:** This option enables the content script to be injected into all URLs, allowing it to perform its implemented functionality.
    - **Resource Inclusions:** The content script references external resources, including tesseract.min.js, MiscUtils.js, sweetaler2.min.js, ModuleLoader.js, and contentScript.js, facilitating functionalities such as OCR, utility functions, alert messages, and content analysis.
    - **Stylesheet Inclusion**The SweetAlert2 stylesheet is included in the content script in order to be successfully loaded and used to create a customized alert message.
5. **Browser Action (Popup):**
    - **Default Popup HTML:** The extension utilizes a default popup HTML file to act as a user interface element.It provides further information about the extension's capabilities.

3.3.2 Content Script: Detection and Real-time Defense Mechanism

The core of our Chrome Extension is the content script, which is in charge of identifying any potentially malicious information included in web pages and offering immediate protection against SSO phishing attacks. This script is crafted to navigate the web page's Document Object Model (DOM) examining different parts and looking for patterns that might be signs of phishing attempts.

Element Traversal Alogirthm

Our first step was defining a function, `traverseElements`, which recursively traverses through the DOM, examining each element encountered. We targeted for inspection a predefined set of HTML tags, including `<button>`, `<a>`, `<span>`, and `<div>`. These elements commonly serve as carriers for phishing content. Upon encountering them, their text content and identifiers (`id` attribute) are scrutinized for potential SSO indicators using regular expressions.
Additionally, image elements need special attention to cover scenarios where phishing content may be embedded within images. To address this, the algorithm includes Optical Character Recognition (OCR) functionality. By analyzing the textual content of the images, the algorithm ensures a comprehensive assessment of possible SSO elements.

Interaction Detection and Message Transmission

After we have located the above elements, then we will have to know when some action took place. Upon interaction with these SSO elements, such as clicking on buttons or links, the listener functions execute, initiating the transmission of messages containing crucial information about the detected content such as the SSO provider.

Message Receival and Background Interaction

In addition to the previous functionality, the content script facilitates communication between the browser environment and the background script, enabling coordinated responses to detect any malicious attempts. Listing 3.2

After receiving messages from the background script, the content script responds accordingly. If the received action is to `confirmRedirect`, implying the need to verify a redirection's legitimacy, the content script assesses the legitimacy status provided. If the redirection is deemed illegitimate, the content script triggers the display of an alert message to notify the user of the potential threat by using the `SweetAlert2` script. Conversely, if the redirection is confirmed as safe, a corresponding message is logged for reference.

```
//Listener Function for receiving messages sent from background script
chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
    if(request.action == "confirmRedirect"){
        if(!request.legit){
            //First load sweet alert and then display a message
            ModuleLoader.loadSweetAlert().then(ModuleLoader.displayAlert());
        }
        else{
            console.log("Safe");
        }
    }
});
```

Listing 3.2: *Receive Validation from Background Script and Alert User*

3.3.3 Utility Class: Miscellaneous Functions

The `MiscUtils` class includes a collection of utility functions for the efficient operation of our Chrome extension. Among these, the `checkIdP` method plays a significant role in identifying potential Single Sign-On indicators using a given attribute. This method takes the provided input and employs a series of regular expression patterns to identify any SSO integration. It operates asynchronously, returning a promise that resolves to either a string representing the detected SSO provider or a boolean value indicating the absence of SSO indicators. Additionally, a regular expression `LENGTH` is utilized to ensure that the name attribute does not exceed a certain length, enhancing efficiency and mitigating potential false positives. Listing 3.3

```
let ssoDetection= {
    FACEBOOK: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|usin
    ↪ g)\s*(facebook|fb)/i,
    GOOGLE: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|using)
    ↪ \s*(google)/i,
    APPLE: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|using)\
    ↪ s*(apple)/i,
    GITHUB: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|using)
    ↪ \s*(github)/i,
    TWITTER: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|using
    ↪ )\s*(twitter)/i,
    MICROSOFT: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|usi
    ↪ ng)\s*(microsoft)/i,
    INSTAGRAM: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|usi
    ↪ ng)\s*(instagram)/i,
    LINKEDIN: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|usin
    ↪ g)\s*(linkedin)/i,
    AMAZON: /(log\s{0,1}(in|on)|sign\s{0,1}(in|up|on)|continue|connect)\s*(with|using)
    ↪ \s*(amazon)/i,
    LENGTH: /^\W*(?:\w+\b\W*){1,10}$/ };
```

Listing 3.3: *Regular Expressions tailored to various SSO providers.*

3.3.4 Module Loader Class: Dynamic Script Loading

This class encapsulates functionality for dynamically loading scripts required for the operation of the extension.

Method: loadSweetAlert

This method is dynamically loading the `SweetAlert2` library into the document. This library is utilized for presenting a customized alert message to users, alerting them of detected phishing attempts. It operates asynchronously, returning a promise after the successful loading of the `SweetAlert2` script.

- **Asynchronous Loading:** This method operates asynchronously, leveraging Promises to ensure seamless integration of the injected library into the extension's environment.
- **Dynamic Script Creation:** After invocation it dynamically creates a `<script>` element, configuring its source attribute to point to the location of the `SweetAlert2` library.
- **Script Injection:** The dynamically created script element is appended to the `<head>` section of the document, initiating the loading process.
- **Promise Resolution:** Upon loading the script with success, the Promise returned by the method resolves, indicating readiness for `SweetAlert2` utilization within the extension.

Method: displayAlert

The `displayAlert` method, Listing 3.4, is responsible for presenting a customized alert message to users. When invoked, it triggers the display of an alert message based on the configuration options that have been provided.

```
static displayAlert(){
    Swal.fire({
        title: "Phishing Attempt Detected",
        text: "Navigation Domain Differs from the expected one",
        icon:'warning',
        confirmButtonText:"Proceed"
    });
}
```

Listing 3.4: *Display Customized Alert Message using SweetAlert2*

3.3.5 Background Script: Navigation and Redirection Handling

The background script of our Chrome extension serves as the central component responsible for handling incoming messages from content scripts and coordinating actions to validate URL redirections against expected domains. This script ensures the

integrity of user navigation by verifying the legitimacy of redirections encountered during interactions with SSO elements.

Message Listener

The background script contains a message listener to receive messages from the content script, specifically targeting messages with the action `checkRedirect`. After receiving such a message, the script proceeds to validate the URL redirection against the expected domain. The listener function orchestrates the validation process by examining the updated URL of the sender tab and comparing it against the expected domain. After manually interacting with legitimate SSO providers we have mapped the expected domain that we should end up in after clicking an SSO element to each provider. Subsequently, a confirmation message is constructed based on the validity of the redirection, which is then sent back to the content script for further action. Listing 3.5

```
// Check if redirection URL domain matches the expected one
if (domain.includes(legit_domain)) {
    message = { action: "confirmRedirect", legit: true };
} else {
    message = { action: "confirmRedirect", legit: false };
}

// Send message back to content script when navigation is completed
chrome.webNavigation.onCompleted.addListener(function navigationListener() {
    // Remove the listener
    chrome.webNavigation.onCompleted.removeListener(navigationListener);
    chrome.tabs.query({ active: true, currentWindow: true }, function(tabs) {
        // Send a message to the content script in the active tab
        chrome.tabs.sendMessage(tabs[0].id, message);
    });
}, { tabId: tabId });
```

Listing 3.5: *Redirection Legitimacy Validation and Message Transmission*

4

Experiment & Results

Our approach centers on addressing the second layer of a phishing attack, distinct from the initial social engineering phase where a malicious actor redirects users to a counterfeit website, ultimately aiming to extract sensitive data. Thus, we bypass the user engagement phase and concentrate on subsequent steps in the process. Eventually, we conducted two distinct experiments to assess our implementation.

In the first part of our experiment, we utilized a phishing toolkit, explained in section 3.1, to create realistic phishing websites. This allowed us to simulate phishing attacks and test the effectiveness of our custom-built Chrome extension. By creating fake login pages with SSO elements for various identity providers, we were able to demonstrate a real-world scenario of a phishing attack and observe the extension's real-time defense capabilities.

The second part of our experiment entailed developing an automation tool to systematically extract malicious domains. These domains were then used as an input into our framework. This large-scale experiment was designed to rigorously assess our framework's performance across a diverse array of websites, trying to detect any existing example of an SSO Phishing attack.

4.1 Real Time Defense Mechanism

In this scenario, our primary objective was to gauge the efficacy of the developed defense mechanism explained in section 3.3 in real-world phishing scenarios. To this end, we seamlessly integrated the implemented browser extension into the experimental setup, thereby enabling users to benefit from real-time protection. Rigorous testing ensued, encompassing diverse scenarios ranging from standard text-based SSO elements to more intricate image-based representations, thereby assessing the extension's adaptability and resilience. Each fake SSO element is illustrated in Figure 3.1. By subjecting the extension to rigorous scrutiny across multiple dimensions, encompassing regular expression-based detection methodologies and Optical Character Recognition (OCR) capabilities, we attained a comprehensive assessment of its robustness and efficacy.

While we conducted these steps for all identity providers, we present here a detailed explanation focusing on one such provider for clarity, specifically Instagram. The

comprehensive series of interactions encountered during the user's engagement with a phishing website is depicted in Figure 4.1.

1. **Redirection to Fake Website:** The user initiates the authentication process and is redirected to a counterfeit website hosting a deceptive login form, indistinguishable from legitimate counterparts.
2. **Selection of SSO Login:** Upon arriving at the fake website, the user opts to log in using Single Sign-On (SSO), a common authentication method prevalent across numerous online platforms.
3. **Redirection to Fraudulent Website:** Subsequently, the user is seamlessly redirected to what appears to be a legit looking website, meticulously crafted using the SocialPhish toolkit, thereby mirroring the interface of the genuine service provider.
4. **Detection of Malicious Activity:** At this critical juncture, our browser extension intervenes, diligently monitoring the user's online activity. Upon detecting subtle signs indicative of a phishing attempt, such as irregular domain patterns or anomalous authentication processes, the extension promptly triggers an alert, notifying the user of the detected malicious activity.

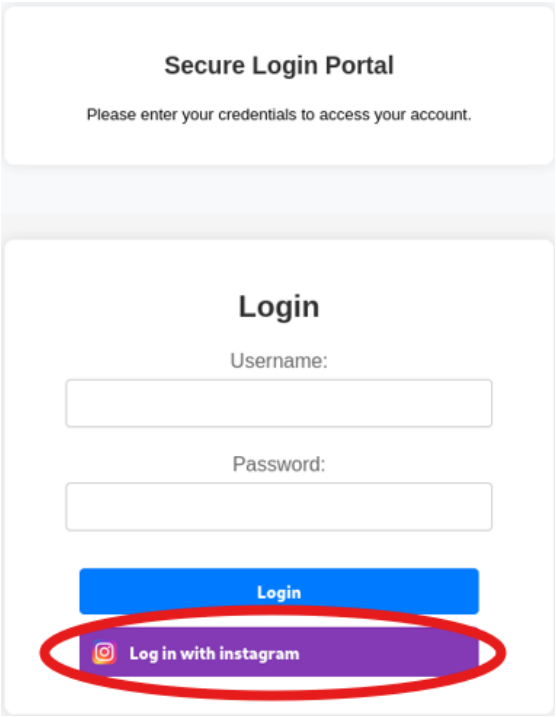
Moreover, the evaluation process extended to validating the extension's functionality against a spectrum of threats, including phishing attempts employing deceptive redirection tactics and diverse authentication mechanisms. By subjecting the extension to rigorous scrutiny across multiple dimensions, we attained a comprehensive assessment of its robustness and efficacy.

Through these various crafted experiments and meticulous evaluations, we gleaned pivotal insights, shedding light on the extension's proficiency in mitigating phishing threats across various modalities. Such empirical evidence serves not only to validate the efficacy of the developed solution but also to underscore its potential as a pivotal component in fortifying users against evolving cybersecurity threats.

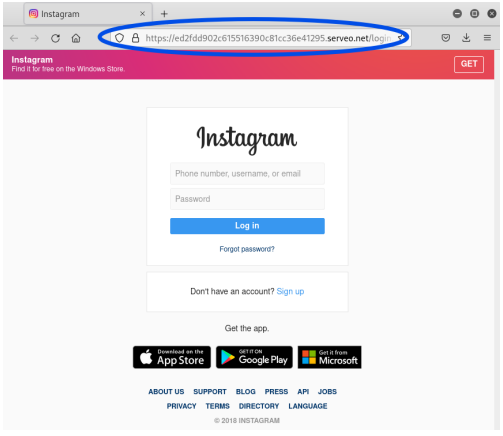
4.2 Large Scale Experiment

In the second part of our experiment, our objective was to evaluate the effectiveness of our automation tool, developed using JavaScript and Puppeteer, presented in section 3.2, on real-life web pages. To achieve this, we needed to conduct a large-scale experiment encompassing malicious websites, trying to detect an instance of the introduced attack vector. This comprehensive approach allowed us to capture a broad spectrum of data, ensuring a thorough assessment of our tool's capabilities.

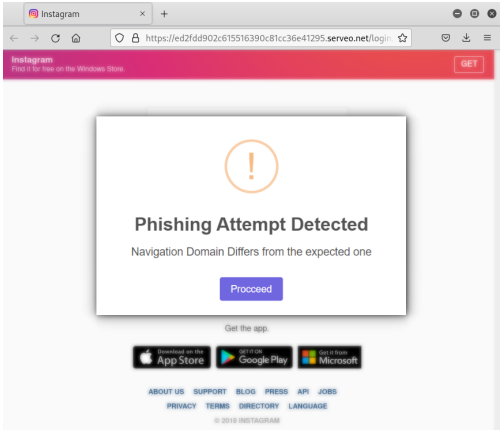
The large-scale experiment was designed to include a diverse range of websites. The inclusion of malicious websites enabled us to log and review metrics specifically related to phishing activities. This approach was essential for a holistic evaluation of the tool's efficacy in identifying malicious patterns during the SSO login.



(a) *User clicks the fake SSO element*



(b) *User gets redirected to fake SSO login page.*



(c) *Defense Mechanism alerts user.*

Figure 4.1: *Illustration of the steps followed by the user during an SSO phishing attack*

4.2.1 Data Collection

The experiment leveraged PhishTank, [25], a renowned collaborative clearing house for phishing data, to source a substantial number of malicious domains. PhishTank, operated by OpenDNS, provides a platform where users can submit, verify, and share information about phishing sites, making it an invaluable resource for our research.

We developed an automation script that systematically visits PhishTank, gathers the registered domains, and inputs them into the automation tool to proceed with the login process. This approach enabled us to efficiently test our tool against a substantial dataset of real-world phishing sites, providing insights into its effectiveness and reliability in detecting and countering phishing attacks. The total number of websites we have used for this large-scale experiment amounts to 10,000. It is worth noting that many websites were not available due to the reports being made. For this reason, to gather the above number we had to run 10,650 web pages.

At phishtank, anyone can submit a phishing page and then it is reviewed to see if the above hypothesis is valid. After analyzing our metrics, we observed that 6,190 of the 10,000 websites were flagged as a valid phish. Furthermore, 343 were flagged as legitimate pages.Table 4.1

Table 4.1: *PhishTank Websites Visited*

Total Websites	% of Valid Phish	% of Invalid Phish	% of Unknown
10,000	61.9%	3.43%	34.67 %

4.2.2 Results

After the large-scale experiment was completed, we analyzed the metrics described in subsection 3.2.2 to gather the results. So we noticed that from the total dataset, the domains that contained any SSO element from the Identity Providers that we included in our research, amount to 312. The above is a percentage of the order of **3.12%**.

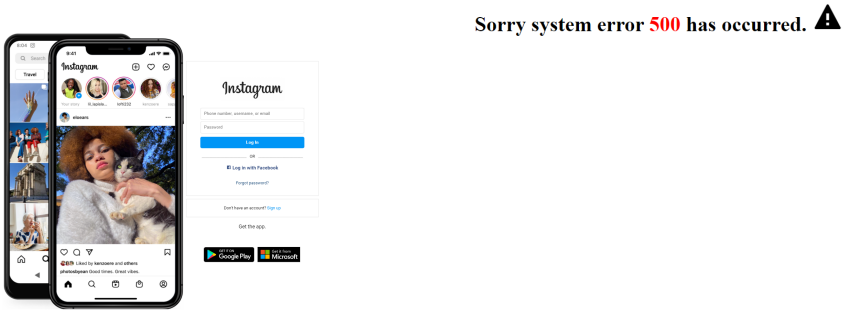
By analyzing more how many times each provider appeared we can draw some conclusions about how common the use of each is. In total, we managed to detect 368 different SSO elements among the visited domains. Google was the most detected Identity Provider, as we identified 130 SSO buttons, covering more than one-third of the total. By observing the Table 4.2 it is clear that Google (35.32%), Facebook (32.06%), and AppleID (19.02%) are the 3 most used providers on the visited sites. Due to the popularity that these providers have, a malicious actor could easily target and exploit the user's trust in them. We managed to locate at least one SSO element from every provider except 2, Amazon and Twitter. To make sure our tool works properly, we opened and checked 100 different websites manually. In this way we verified if we had false positive or false negative measurements during the experiment. Finally, it turned out that all the SSO elements detected were legitimate SSO buttons and moreover there

was none that stayed undetected by the framework.

Table 4.2: *Detected Single Sign-On Elements*

Identity Provider	Occurrences	% of Total
Google	130	35.32 %
Facebook	118	32.06 %
AppleID	70	19.02 %
Microsoft	21	5.70 %
Github	15	4.07 %
LinkedIn	10	2.71 %
Instagram	4	1.08 %
Amazon	0	0.00 %
Twitter	0	0.00 %

As previously mentioned, one goal of this research was to investigate whether there are any examples of phishing attacks on SSO in the real world. In this particular dataset, we could not find such an example, but it is worth highlighting a specific case. In this instance, an SSO element, specifically Facebook's, was detected, and instead of being directed to the expected login page, we were redirected to a blank page. Even though this does not indicate a clear case of a phishing attack, it serves as a sign of a potentially malicious pattern, Figure 4.2.



(a) *Malicious Domain with SSO Element* **(b)** *Redirection Site*

Figure 4.2: *Malicious Pattern Detected*

Studying the metrics kept for the page, from the `HTML` code and the screenshot taken, we identify that it is an impersonated page of the Instagram login page, as illustrated in Figure 4.2a. Furthermore, the code contains the link to which the Facebook SSO element redirects us, and upon observing the URL, Listing 4.1, we detect that it is different from the expected one. Although in the next step, there is no form requesting our login credentials for Facebook SSO, what appears is shown in Figure 4.2b.

This redirection to a different page from the legitimate one indeed constitutes a malicious pattern and cannot go unnoticed. However, on its own, it is not sufficient to

```
1 .
2 <a href="error.html" class="fb-login" target="_blank"><i class="bx
  ↪  bxl-facebook-square"> </i> Log in with Facebook</a>
3 .
```

Listing 4.1: *Code snippet from metrics*

be considered a complete phishing attack as defined in section 3.1.

Another interesting metric is the nature of the SSO elements found during our experiment. We came across different kinds of implementations such us buttons, images, anchor tags etc. By extracting the elements from the HTML codes of each page we can observe which type is used the most. Out of the 368, 171 consisted of buttons, 90 of anchor tags and 37 of images. Making these elements the 3 most frequently appearing. Table 4.3

Table 4.3: *Different Types of SSO Elements Found*

SSO Elements	Occurrences	% of Total
Buttons	172	46.73%
Anchor Tags	96	26.08%
Images	37	10.05%
Div Container	32	8.69%
(Spans, Inputs, etc.)	31	8.42%

By completing this large scale experiment we were able to drill down and review malicious websites for our purpose. We noticed whether sso elements are included in them, the providers that are used the most but also the different types that such an element can have.

5

Related Work

In this section, we present a survey of relevant literature addressing themes akin to our research focus. Our exploration encompasses seminal and contemporary studies in the field of cybersecurity, specifically concerning the assessment and mitigation of phishing attacks. Through this review, we aim to situate our work within the broader scholarly discourse, drawing upon established methodologies and insights. By scrutinizing existing research, we endeavor to discern patterns, identify gaps, and refine our approach.

5.1 Threat Assessment: Evaluating Phishing Attacks

A study made by [32] was based on exploring the feasibility of creating extreme phishing attacks that have an almost identical look and feel to those of the targeted websites, and evaluating the effectiveness of such phishing attacks. Their user study results based on 194 participants demonstrate that phishing attacks constructed by their toolkit are highly effective. The questionnaire results showed that **178 (91.8%)** of the 194 participants were not suspicious of the extreme phishing websites that they visited. Observation results show that **182 (93.8%)** of the 194 participants submitted their credentials. In the meanwhile, most of the "victims" were aware of phishing before they participated in this study. Therefore, it is reasonable to assume that phishing attacks will be widely adopted adopted deployed increasingly in the future. Also, a a comprehensive review of recent advancements in phishing detection techniques, aiming to bridge existing gaps and enhance understanding was presented by [1]. By analyzing literature from the past four years, the study elucidates current trends and challenges in online phishing detection, providing valuable insights for future research and development efforts. The study identifies social media phishing as a major domain within this landscape, which was a major reason that led us to conduct our presented research.

5.2 Detecting Phishing Websites

A relevant research [4] utilizes adversarial deep learning to simulate the behavior of malicious actors and detect phishing websites. By training a generative adversarial

network (GAN) on both legitimate and phishing website data, DeepPhish effectively discerns subtle visual cues indicative of phishing attempts, thus enhancing detection accuracy. Its approach is particularly effective in identifying sophisticated phishing websites that mimic legitimate ones, thereby providing a valuable tool for cybersecurity practitioners. A novel method for detecting phishing websites based on the analysis of Document Object Model (DOM) tree structures, which mirrors the methodology adopted in our research was proposed by [18]. By leveraging features extracted from the DOM trees of websites, the proposed approach achieves robust and accurate detection of phishing attempts, outperforming traditional URL-based methods. Its reliance on the structural characteristics of web pages offers a promising avenue for detecting phishing attempts that evade conventional detection techniques. Furthermore, the escalating threat of web phishing attacks by proposing hybrid intelligent phishing website prediction techniques is addressed on [2]. Conventional methods relying on blacklists often fall short in accurately identifying evolving phishing websites. Leveraging deep neural networks (DNNs) alongside evolutionary algorithm-based feature selection and weighting, the proposed approach enhances prediction accuracy. Through genetic algorithms (GA), influential features and optimal weights are determined, empowering DNNs to accurately predict phishing websites. Experimental results showcase superior classification accuracy, sensitivity, specificity, and geometric mean compared to existing methods, affirming the efficacy of the proposed hybrid intelligent approach.

5.3 Single Sign-On significant role in user's security

A review made by [6] pointed out some problems that may arise during the use of SSO as stealing the user's information the malicious user having great power in his hands. They analyzed the benefits and drawbacks of SSO usage and also the combination of SSO with Multi-Factor Authentication. Also, related work [14] presents a study of 208 popular sites in the Tranco top 5K that support account creation to understand the availability of MFA and RBA on the web, the additional authentication factors that can be used for MFA and RBA, and how logging into sites through more secure SSO providers change the landscape of user authentication security. Their results show that only 42.31% of sites support any form of MFA, and only 22.12% of sites block an obvious account hijacking attempt but SSO completely changes the picture. They found that 167 sites (80.29%) in their dataset either have MFA or could inherit it through SSO providers. Additionally, 161 sites (77.40%) in our set either use RBA or could inherit it through SSO providers. Finally, 151 sites (72.60%) have or inherit an RBA mechanism that blocks the suspicious login attempt.

As the paper suggests, attackers may leverage users' trust in SSO to carry out phishing attempts, underscoring the importance of enhancing security awareness and implementing effective countermeasures. Furthermore, a relevant research [26] was focused on presenting a different approach for efficient and secure single sign-on frameworks based on proxy signature schemes. They made an analysis and a comparison of previous

architectures against the advantages and disadvantages of each kind. They managed to provide a transparent single sign-on without undermining network security and without requiring any online communications between service providers and identity providers.

5.4 Single Sign-On in Cloud Computing

A relevant study made by [21] has designed a single sign on for cloud users. They propose a solution with established standards for open authorization as we have studied in our research. Another Single Sign-On implementation was designed by [5] that enables users of an organization to seamlessly access various applications without re-entering the credentials. They used AWS Services and OKTA to provide seamless navigation among various websites. Because cloud services are becoming more and more popular in the IT sector, it is imperative that their operations be kept secure.

5.5 Gathering a Comprehensive Dataset

A research was based on a category similar to phishing. One of the research's goals was to create a classifier that would detect fraudulent e-commerce websites (FCWs). [7].
A major difficulty that they faced was the lack of an existing up-to-date, clean dataset of FCWs being publicly available. Curating such a list was a challenging task as suck websites disappear quickly. That is the same issue that we faced while trying to run a large-scale experiment to test our automation tool.
To address this challenge they took advantage of Reddit (social media platform), with approximately 330 million users [13] discussing various topics. They collected a dataset of users' submissions and the corresponding comments based on a thread dedicated to discussing FCWs. The data that they extracted from each URL helped them gather resources and features. These features were used to create their classifier called Beyond Phish.

6

Discussion & Future Work

6.1 Discussion

The culmination of our research endeavors heralds a significant milestone in the ongoing battle against cyber threats, particularly in the domain of automating Single Sign-On (SSO) detection and bolstering defenses against phishing attempts. Through meticulous analysis and experimentation, we encountered some of the myriad ways in which malicious actors exploit wide adopted systems like SSO and leverage sophisticated toolkits to perpetuate their nefarious activities with alarming ease.

We have developed and put into place an automation framework that can carry out extensive experiments to discover cases of phishing during SSO login procedures in response to these constantly changing threats. Our framework utilizes technologies and processes to provide users with the necessary tools and insights to prevent future security breaches and protect their digital identities.

Our strategy's key component is the incorporation of a real-time defensive mechanism, which gives users proactive online protection against phishing attacks. With the help of this cutting-edge technology, users may become more resilient and aware of any risks in real time, which reduces the likelihood that they will fall victim to malicious schemes.

Furthermore, as part of our comprehensive validation process, we assumed the role of malicious actors, utilizing the implemented toolkits to create phishing websites and simulate real-world attack scenarios. Through the use of a hands-on approach, we were able to conduct a realistic demonstration of our extension's capabilities in a live setting, showing its accuracy and reliability in identifying and preventing phishing attacks. We were able to show our framework's effectiveness and illustrate its potential to be an essential protection mechanism against online dangers through these real-world simulations.

In the future, our efforts will hopefully act as a catalyst for more innovation and cooperation in the cybersecurity area. We stay steadfast in our commitment to continuous research and development even as dangers continue to change and multiply. We are prepared to further improve our framework's relevance and efficacy through ongoing optimization, collaboration with industry partners, and refinement. This will guarantee

that our approach continues to have an influence on protecting digital identities and promoting trust in online interactions. Still, there's a lot of space for improvement, especially in terms of effectiveness and usefulness.

6.2 Directions for Future Work

In the preceding section, we elucidated our research efforts and expounded upon the implementation of our framework, including its strengths and acknowledged limitations. Our exposition provided a thorough overview of our methodology's current state, highlighting both its capabilities and areas necessitating further refinement. As we conscientiously identified specific points requiring additional scrutiny, we now transition to outlining potential avenues for future research and development. This forthcoming section aims to meticulously examine areas for enhancement, guided by our commitment to advancing cybersecurity measures effectively. Through this exploration, we endeavor to fortify the resilience of our framework against evolving threats and ensure its continued relevance in safeguarding digital identities.

Future research endeavors could delve into expanding the spectrum of interactions available on websites. While the implemented tool adeptly traverses websites to ascertain the presence of SSO buttons, augmenting its capabilities to encompass a broader array of features would be advantageous. Such enhancements would facilitate a more comprehensive exploration of the security landscape inherent to these platforms. By engaging with diverse authorization prompts, session management functionalities, and authentication techniques, the system could furnish a more nuanced evaluation of potential vulnerabilities and corresponding countermeasures.

Additionally, bolstering the roster of SSO providers - presently standing at nine - would enhance the system's adaptability and utility across a wider array of online services. Augmenting the evaluation framework with additional Identity Providers (IdPs) not only enhances the dataset's quality but also engenders a deeper understanding of the multifaceted tactics and strategies underpinning SSO integration across diverse platforms.

The availability and caliber of datasets featuring malicious sites employing phishing SSO implementations are pivotal for further advancement. While the current study leverages pre-existing datasets, the system's efficacy could be substantially heightened by access to a more extensive and up-to-date repository of identified malicious sites. Given the ephemeral nature of these websites, promptly identified and removed following user reports, maintaining an accurate dataset poses a significant challenge. However, cooperative initiatives with cybersecurity organizations, sustained monitoring endeavors, and concerted data-sharing initiatives could facilitate access to pertinent and timely information, thereby fortifying the system's capacity to identify and mitigate emerging threats.

Furthermore, overcoming well-known obstacles including the dynamic nature of online environments, the spread of advanced phishing techniques, and the growth of

attack vectors will have a significant impact on future research endeavors. Utilizing methods like natural language processing and machine learning could improve the system's adaptiveness and predictiveness, making it more capable of protecting off changing threats and anticipating hostile actions. The initiative to develop a Deep Learning-based method for visually identifying phishing web pages, as described in [19], highlights the possible benefits that can arise from combining components from several research projects.

In conclusion, in this research we demonstrated an attack which targets a wide adopted technology like Single Sign-On. Our implementation provided a way in which an attacker can steal user's credentials while he is trying to login using a legitimate provider. We have created an automation framework to detect any existing threat of this kind by conducting large scale experiment on various malicious domains. Eventually we provide a defense safeguard to users by the design of a Chrome Extension. This tool can be loaded into user's browser and provide real-time prevention of a phishing attempt. While the implemented tools represent a significant stride towards automating SSO detection and phishing prevention, ongoing research and development endeavors are imperative to ensure its continued efficacy in the ever-evolving cybersecurity landscape. Through continued refinement, optimization, and collaborative engagement with pertinent stakeholders, the system holds the promise of emerging as a formidable bulwark safeguarding users against online identity theft and unauthorized access.

This page intentionally left blank.

Bibliography

[1] Mafaz Alanezi. "Phishing Detection Methods: A Review". In: *Technium: Romanian Journal of Applied Sciences and Technology* 3 (Nov. 2021), pp. 19–35. DOI: 10.47577/technium.v3i9.4973.

[2] Waleed Ali and Adel A. Ahmed. "Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting". In: *IET Information Security* 13.6 (2019), pp. 659–669. DOI: https://doi.org/10.1049/iet-ifs.2019.0006. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ifs.2019.0006. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ifs.2019.0006.

[3] Calvin Ardi and John Heidemann. "AuntieTuna: Personalized Content-based Phishing Detection". In: 3. Jan. 2016. DOI: 10.14722/usec.2016.23012.

[4] Alejandro Correa Bahnsen et al. "DeepPhish : Simulating Malicious AI". In: 2018. URL: https://api.semanticscholar.org/CorpusID:51691528.

[5] Dr.Kachapuram BasavaRaju. "Single Sign on Using Cloud Computing". In: *Cybersecurity* (2019). URL: https://api.semanticscholar.org/CorpusID:214665898.

[6] Tayibia Bazaz and Aqeel Khalique. "A Review on Single Sign on Enabling Technologies and Protocols". In: *International Journal of Computer Applications* 151 (Oct. 2016), pp. 18–25. DOI: 10.5120/ijca2016911938.

[7] Marzieh Bitaab et al. "Beyond Phish: Toward Detecting Fraudulent e-Commerce Websites at Scale". In: *2023 IEEE Symposium on Security and Privacy (SP)*. 2023, pp. 2566–2583. DOI: 10.1109/SP46215.2023.10179461.

[8] Mike Britton. "Microsoft Impersonated Most in Phishing Attacks Among Nearly 350 Brands". In: *Abnormal Blog* (2023). URL: https://abnormalsecurity.com/blog/credential-phishing-trends-2023.

[9] *Chromium*. [Online]. 2024. URL: https://chromium.googlesource.com/chromium/src/+/HEAD/docs/accessibility/overview.md.

[10] Cloudflare. *What is Pishing Attack*. 2024. URL: https://www.cloudflare.com/learning/access-management/phishing-attack/.

[11] Debendra0256. "Develop Your First Google Chrome Extension Using HTML and JQuery". In: *Articles* (2017). URL: https://www.codeproject.com/Articles/1185723/Develop-Your-First-Google-Chrome-Extension-Using-H.

[12] Tarique Ejaz. "Chrome Extension Tutorial: How to Pass Messages from a Page's Context". In: *freeCodeCamp* (2021). URL: https://www.freecodecamp.org/news/chrome-extension-message-passing-essentials/.

[13] F.Team. "Reddit Content Stats(Demographics, Usage Traffic Data)". In: *a* (2024). URL: https://foundationinc.co/lab/reddit-statistics/.

[14] Anthony Gavazzi et al. "A study of multi-factor and risk-based authentication availability". In: *Proceedings of the 32nd USENIX Conference on Security Symposium*. SEC '23. Anaheim, CA, USA: USENIX Association, 2023. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/gavazzi.

[15] Mohammad Ghasemisharif, Chris Kanich, and Jason Polakis. "Towards Automated Auditing for Account and Session Management Flaws in Single Sign-On Deployments". In: *2022 IEEE Symposium on Security and Privacy (SP)*. 2022, pp. 1774–1790. DOI: 10.1109/SP46214.2022.9833753.

[16] Google. *Puppeteer: Headless Chrome Node.js API*. https://github.com/puppeteer/puppeteer. 2024.

[17] Darren Guccione. "Password Fatigue Is Real". In: (2023). URL: https://biztechmagazine.com/article/2023/02/password-fatigue-real-heres-what-businesses-need-know.

[18] Dac-Nhuong Le. "Detecting Phishing Web Pages based on DOM-Tree Structure and Graph Matching Algorithm". In: Dec. 2014. DOI: 10.1145/2676585.2676596.

[19] Yun Lin et al. "Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3793–3810. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/lin.

[20] Samuel Marchal and N. Asokan. "On Designing and Evaluating Phishing Webpage Detection Techniques for the Real World". In: *11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18)*. 4. Baltimore, MD: USENIX Association, Aug. 2018. URL: https://www.usenix.org/conference/cset18/presentation/marchal.

[21] Pratap Murukutla and K.C. Shet. "Single Sign on for Cloud". In: *2012 International Conference on Computing Sciences*. 2012, pp. 176–179. DOI: 10.1109/ICCS.2012.66.

[22] Adam Oest et al. "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis". In: 1. May 2018, pp. 1–12. DOI: 10.1109/ECRIME.2018.8376206.

[23] Adam Oest et al. "PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists". In: *29th USENIX Security Symposium (USENIX Security 20)*. 5. USENIX Association, Aug. 2020, pp. 379–396. ISBN: 978-1-939133-17-5. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/oest-phishtime.

[24] OneLogin. "How Does Single Sign-On Work?" In: (2024). URL: https://www.onelogin.com/learn/how-single-sign-on-works.

[25] OpenDNS. *PhishTank*. n.d. URL: https://www.phishtank.com/.

[26] Anita Patil and Rakesh Pandit. "Analysis of Single Sign on for Multiple Web Applications". In: (Aug. 2013). URL: https://www.ijareeie.com/upload/2013/august/20S_Analysis.pdf.

[27] Yaqoob Al-Slais and Wael El-Medany. "User-Centric Adaptive Password Policies to Combat Password Fatigue". In: *The International Arab Journal of Information Technology* (Jan. 2022). DOI: 10.34028/iajit/19/1/7.

[28] *SocialPhish*. [Online]. 2024. URL: https://github.com/TYehan/SocialPhish/blob/main/README.md.

[29] Statista. *Most Targeted Phishing Industries 2023*. 2024. URL: https://www.statista.com/statistics/420442/organizations-most-affected-by-phishing-by-industry/.

[30] Taina Teravainen. *What is Single Sign-On (SSO) and How Does It Work?* Accessed: 2024-05-09. 2024. URL: https://www.techtarget.com/searchsecurity/definition/single-sign-on.

[31] Guang Xiang et al. "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites". In: *ACM Trans. Inf. Syst. Secur.* 14 (Sept. 2011). 2, p. 21. DOI: 10.1145/2019599.2019606.

[32] Rui Zhao et al. "Design and evaluation of the highly insidious extreme phishing attacks". In: *Comput. Secur.* 70 (2017), pp. 634–647. DOI: 10.1016/j.cose.2017.08.008.

This page intentionally left blank.

Appendices

This page intentionally left blank.

A

Web Accessibility API

Listing A.1 includes an HTML code sample for a login page, which contains two text boxes, for the username and password, a submit button, and Facebook's SSO button. Listing A.2 depicts the accessibility tree obtained from Chromium's Web Accessibility API upon visiting the page. In this research, the Puppeteer's accessibility API was modified in order to include the node information that was not included in the accessibility tree to map each object back to its DOM element. The accessibility tree only contains simplified information, which includes a subset of HTML elements that are deemed to be useful (e.g., it does not include the hidden input element). For the purpose of SSO detection, they applied a set of regular expressions on the accessibility tree values to find potential candidates for SSO buttons, and username and password fields related to their test workflows.
While observing the last node in Listing A.2, we conclude that it is a possible SSO element that can initiate a login process.

```
<form method="get" action="/submit" validate>
    <h3>Login Form</h3>
    <div>
        <label for="username">Email</label>
        <input type="text" name="email" required>
        <label for="password">Password</label>
        <input type="password" name="password" required>
        <input type="hidden" id="test" name="secret" value="1234">
    </div>
    <div>
        <button type="submit">Login</button>
    </div>
    <fb:login-button scope="public_profile,email" onlogin="checkLoginState();">
    </fb:login-button>
 </form>
```

Listing A.1: *Sample HTML code for a login page.*

```
{
    "role": "WebArea",
    "name": "",
    "children": [
    {
        "role": "heading",
        "name": "Login Form",
        "level": 3
    },
    {
        "role": "text",
        "name": "Email"
    },
    {
        "role": "textbox",
        "name": "",
        "required": true,
        "invalid": "true"
    },
    {
        "role": "text",
        "name": "Password"
    },
    {
        "role": "textbox",
        "name": "",
        "required": true,
        "invalid": "true"
    },
    {
        "role": "button",
        "name": "Login"
    },
    {
        "role": "Iframe",
        "name": "fb:login_button Facebook Social Plugin"
    }]
}
```

Listing A.2: *Accessibility tree obtained for Web Accessibility API.*

B

Detailed Explanation of Socialphish

This appendix provides a detailed overview of Socialphish and its usage, as an advanced phishing tool that automates the creation of phishing websites to impersonate popular brands. It is widely used for educational purposes and security testing to demonstrate the risks and methods of phishing attacks. This tool available on GitHub, a platform for collaborative software development. To acquire SocialPhish, you can download the repository directly from its GitHub page. Another way, that gives you a local copy of SocialPhish on your computer, is by cloning the GitHub Repository. With this method, you are actually running the most recent version of the program and you are taking advantage of any improvements or bug fixes that the developers have made, with a great convenience.

B.1 Usage

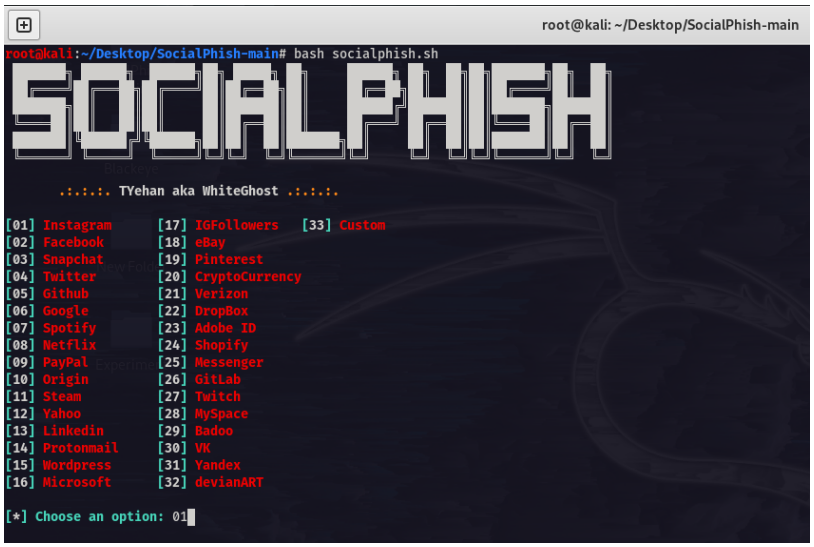
Once installed, Socialphish provides an interface for selecting and creating phishing websites. With the usage of this program, you may mimic numerous well-known internet services, like Google, Facebook, and many more. This is helpful for examining and showcasing the various platforms on which phishing assaults can be carried out.

1. **Launch SocialPhish:** Upon running the setup script, Socialphish launches and presents a menu of options. Each option corresponds to a different website template that can be used for phishing.
2. **Select Brand Template:** From the list of available templates, you select the wanted brand by entering the corresponding number. This action tells SocialPhish to prepare a phishing page that mimics IdP's login interface.
3. **Configuring Port and Mask URL:** SocialPhish prompts you to enter a port number on which the phishing site will be hosted. To increase the credibility of the phishing attempt, you can optionally provide a mask URL. This stage is essential for giving the phishing page the appearance of legitimacy to the unwary user.
4. **Generating and Sharing the Link:** After following the previous steps, SocialPhish generates a URL that leads to the phishing page. This URL can be distributed

to potential targets, who will see a login page that looks identical to Facebook's official login page.

5. **Capturing Credentials:** This feature illustrates how quickly sensitive information may be obtained, highlighting the tool's capacity to exhibit the entire impact of phishing attempts.

B.2 Screenshots and Demonstrations



(a) *SocialPhish Menu*

(b) *Select Template and Port Forwarding*

(c) *Capture victim's Agent Information Credentials*

Figure B.1: *Demonstration of Socialphish creating a phishing website, highlighting template selection, configuration, link generation, and credential capture.*

B.3 Ethical Considerations and Usage

The fact that Socialphish should only be utilized for ethically questionable objectives—like security testing, instructional demonstrations, and increasing cybersecurity awareness—must be emphasized. Phishing tool usage without authorization can have serious legal repercussions as well as ethical transgressions. Thus, before carrying out any phishing simulations, be sure you have the appropriate authorization.