One Click to Leak: Characterizing the Real-World Usage and Threat Impact of MNO-based Single Sign-On Websites

Jiasheng Huang*
Tsinghua University
Beijing, China
huangjc21@mails.tsinghua.edu.cn

Mingxuan Liu*
Zhongguancun Laboratory
Beijing, China
liumx@mail.zgclab.edu.cn

Pei Chen
Fudan University
Shanghai, China
peichen19@fudan.edu.cn

Baojun Liu✉†
Tsinghua University
Beijing, China
lbj@tsinghua.edu.cn

Yiming Zhang
Tsinghua University
Beijing, China
zhangyiming@tsinghua.edu.cn

Geng Hong
Fudan University
Shanghai, China
ghong@fudan.edu.cn

Zhenrui Zhang
Baidu Inc.
Beijing, China
zhangzhenrui@baidu.com

Hai Yang
Baidu Inc.
Beijing, China
yanghai01@baidu.com

Haixin Duan‡
Quancheng Laboratory
Jinan, China
duanhx@tsinghua.edu.cn

Hui Jiang✉†§
Tsinghua University
Beijing, China
jianghui01@baidu.com

Abstract

Mobile Network Operator (MNO)-based Single Sign-On (MSSO) is a password-free authentication framework that relies on mobile data sessions. Unlike traditional SSO, it shifts the Identity Provider (IdP) to the MNO and the authentication anchor to the Service Provider (SP). MSSO is increasingly deployed and has expanded from mobile apps to websites. However, the emerging ecosystem and security risks of web-based MSSO remain largely unexplored.

To fill this gap, we first analyze mainstream MSSO deployments and identify a 3-phase workflow with three trust defects enabling the trust hijacking threat. We further demonstrate *One-Click-to-Leak (OCL)* attacks, where a single webpage visit can leak sensitive identity information (e.g., phone numbers). To systematically assess real-world deployment and risk, we conduct the first large-scale, longitudinal study of web-based MSSO with a leading security company. We design a hierarchical detection framework leveraging passive DNS correlations and URL reconstruction from search data to identify MSSO-enabled websites. Over one year, we identified 116,852 such website URLs across 729 apex domains. Unfortunately, 73.6% of these URLs exhibit at least one trust defect: 69.4% expose developer credentials, and 27.1% issue high-privilege tokens before user consent, showing that OCL-enabling trust defects are widespread in real deployments. Structurally, 31.8% of the 729 apex domains rely on Resellers, obscuring the downstream SP from the MNO in the analyzed flows. By analyzing scripts on MSSO-enabled sites, we identify 101 websites strongly associated with OCL attack behavior. In collaboration with our partner, we trace a representative upstream platform that was subsequently seized by law enforcement and uncover a monetized underground ecosystem. Sanitized backend data shows that it collected 14,100 users' phone numbers within three days and linked them to sensitive information such as browsing activity. Our work provides a comprehensive study of the real-world deployment landscape and security implications of web-based MSSO. Through responsible disclosure, our work helps secure the mobile authentication ecosystem.

CCS Concepts

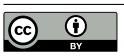
• **Security and privacy** → **Authentication**; **Web application security**; • **General and reference** → *Measurement*.

Keywords

MNO-based Single Sign-On, One-Click-to-Leak, privacy leakage

ACM Reference Format:
Jiasheng Huang, Mingxuan Liu, Pei Chen, Baojun Liu✉, Yiming Zhang, Geng Hong, Zhenrui Zhang, Hai Yang, Haixin Duan, and Hui Jiang✉. 2026. One Click to Leak: Characterizing the Real-World Usage and Threat Impact of MNO-based Single Sign-On Websites. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3830454.3846782

*Both authors contributed equally to this work.
†✉ Corresponding authors.
‡Also with Tsinghua University.
§Also with Baidu Inc..



CCS '26, The Hague, Netherlands

ACM ISBN 979-8-4007-2871-6/2026/11
https://doi.org/10.1145/3830454.3846782

Figure 1: Examples of Web-Based MSSO.

1 Introduction

Mobile Network Operator (MNO)-based Single Sign-On (MSSO) has emerged as a password-free authentication framework that uses a user's active mobile data session (e.g., 4G/5G) as an identity signal, also termed one-click login. In an MSSO flow, the MNO acts as the Identity Provider (IdP): it maps the user's mobile data session to a verified subscriber identity, typically a phone number, and allows a Service Provider (SP) to authenticate the user through a one-click login flow, as shown in Figure 1. A similar delegation pattern appears in Web SSO, although its trust anchor and identity source differ: an independent Web IdP authenticates an IdP account and asserts identity to the SP through browser redirects. By reducing password entry, manual SMS OTP retrieval, and the browser redirects required by conventional Web SSO, MSSO offers a low-friction authentication experience for login, onboarding, and account verification.

Originally deployed in native mobile applications, MSSO is now expanding onto the open web, allowing ordinary webpages to initiate this one-click authentication flow. Within MSSO, this delegated relationship is implemented differently across client environments: native applications encapsulate the authentication logic in proprietary SDKs and compiled binaries, whereas web deployments execute it as JavaScript in the browser, where authentication parameters, tokens, and developer credentials may appear in readable code. However, this expansion exposes MSSO's delegated trust model to the open architecture of the web. Prior work has examined account and session security in Web SSO and weaknesses in native one-tap authentication [14, 17, 49], but it does not fully characterize how MSSO's delegated trust boundaries are established and enforced when authentication logic runs in the browser.

Empirical Study: Three-phase Workflow. To systematically investigate web-based MSSO, we first reconstruct the real-world deployment and operation of it via empirical analysis. By analyzing provider documentation and real-world deployments, we confirm 14 MSSO Providers that typically offer services through APIs or JavaScript SDKs, and identify two roles: direct MNO providers, which perform network-level identity verification, and Third-Party Resellers, which rebundle MNO authentication capabilities through unified developer-facing interfaces. We further reconstruct a common three-phase workflow of web-based MSSO: *Pre-authentication*, *Token Issuance*, and *Backend Verification*. This workflow captures how the delegated trust chain operates on the web and provides the basis for analyzing where it breaks.

Trust Hijack Threat. Furthermore, we uncover three *Trust Defects* in this delegated trust chain. *Trust Defect 1: Unenforced User Consent* arises because the MNO cannot verify at the protocol level that the user approved an authentication request. *Trust Defect 2: Static Credential Exposure* stems from the use of static SP developer credentials in readable web assets, where exposure undermines the credential-based boundary for authenticating legitimate SPs. *Trust Defect 3: Origin Validation Blindspot* results from the MNO's inability to reliably identify the true request origin, especially when Third-Party Resellers mediate authentication. Together, these defects break the delegated trust chain and enable a broader *Trust Hijack Threat*, in which an adversary hijacks the trust relationships embedded in MSSO. Its concrete attack form is *One-Click-to-Leak (OCL)*: a single webpage visit over a mobile data session can leak the user's verified phone number without an explicit login action.

Longitudinal Real-World Measurement. Measuring this threat at web scale is challenging. MSSO integrations are sparse and distributed, and OCL-relevant behavior appears in authentication traffic and JavaScript execution rather than static page content alone. Brute-force crawling is computationally inefficient, while monitoring real users' authentication traffic would be costly and privacy-invasive. Therefore, we collaborated with a leading search engine company, Baidu, to design and implement OCL Detector, a scalable measurement framework with a staged pipeline: it uses passive DNS correlation to narrow the broader web to domains associated with MSSO authentication, maps these domains to candidate URLs using search-engine data, verifies MSSO integrations through targeted dynamic probing in a simulated mobile environment, and applies script-level abuse analysis to identify high-risk MSSO usage.

Using OCL Detector, we conduct a one-year measurement of web-based MSSO deployments. We identify 116,852 MSSO-enabled URLs across 729 apex domains, spanning high-trust sectors such as finance, e-commerce, and travel. *The results show that Trust Defects are widespread: 73.6% of these URLs exhibit at least one Trust Defect, 69.4% expose developer credentials, and 27.1% issue high-privilege tokens before user consent.* At the domain level, 31.8% of the 729 apex domains rely on Third-Party Resellers, leaving MNOs structurally blind to the true request origin. These findings show that Trust Defects are prevalent across the web-based MSSO ecosystem rather than isolated deployment mistakes.

Beyond prevalence, we show that OCL is already exploited in the wild. By analyzing scripts that trigger MSSO calls, we identify 101 websites strongly associated with abusive MSSO exploitation through scripts supplied by 5 upstream platforms. Through collaboration with an industry partner, our findings supported a law-enforcement investigation that led to the seizure of several abusive platforms. We analyze sanitized backend code and data from one representative seized platform. The seized platform collected 14,100 unique phone numbers within three days and linked these identifiers to sensitive web context such as browsing activity and search keywords, revealing a monetized pipeline for large-scale user de-anonymization.

Contribution. This paper makes the following contributions:

- *Web-based MSSO Characterization:* Through systematic empirical study, we characterize the provider landscape, delegated trust model, and common three-phase workflow of MSSO.

• *Novel Trust Hijack Threat:* We identify three Trust Defects in web-based MSSO deployments that jointly enable the *Trust Hijack Threat*. We further identify *One-Click-to-Leak (OCL)* as a concrete attack, leaking users' verified phone numbers through simple mobile-data webpage access without explicit login.
• *Large-Scale Measurement:* We design OCL Detector, a scalable measurement framework. Combining passive DNS correlation and search engine perspectives, we conduct large-scale measurements on real-world MSSO deployments and underlying trust defects. Over one year, we identify 116,852 MSSO-enabled URLs across 729 apex domains, and quantify that 73.6% of web-based MSSO deployments exhibit at least one trust defect. We further pinpoint 101 abuse-related websites linked to OCL exploitation via five upstream platforms. Analysis of anonymized backend data from a representative takedown service revealed 14,100 collected phone numbers within three days, highlighting the significant risks of OCL.
• *Defense Insight and Disclosure:* To mitigate the identified risks, we propose practical defense mechanisms. We also conduct responsible disclosure to affected MNOs, SPs, and law-enforcement agencies and provide active assistance with our collaborators. Among the domains identified by OCL Detector, we observed that at least 148 affected sites had deployed recommended defenses, including safer MSSO implementations such as masked-number completion.

2 MNO-based SSO Preliminaries

In this section, starting by illustrating the evolution of web authentication, we introduce the Mobile Network Operator (MNO)-based single sign-on (MSSO) and its landscape.

2.1 Evolution of Web Authentication

User authentication serves as the critical line of defense for account security. To better balance security and usability in the authentication process, mechanisms have evolved from peer-to-peer schemes (e.g., password-based login) toward trust-delegated authentication frameworks (e.g., Web Single Sign-On). Figure 2 summarizes this progression through four representative authentication schemes.

Password-based authentication treats the user as the sole trust anchor, requiring the user to memorize and input a pre-negotiated secret into the login interfaces of the Service Provider (SP) [8, 9]. To reduce the memorization overhead of passwords, SMS One-Time Passwords (OTP) introduce an MNO-assisted verification step [33, 36, 37]. This mechanism relies on MNOs to verify the user's identity via phone number–subscriber binding and issue OTPs as login credentials. To improve usability, the web ecosystem adopts Web Single Sign-On (SSO) (e.g., OAuth 2.0) [2, 24, 34, 41]. Web SSO delegates trust to centralized Identity Providers (IdPs, e.g., Google [18]), which authenticate users and assert identity to Service Providers via browser redirects. However, Web SSO requires active IdP sessions; otherwise, users must undergo a multi-step login or registration process, increasing friction.

MNO-based SSO (MSSO) (also termed one-tap authentication) provides a streamlined login experience [20]. Despite relying on a similar trust delegation mechanism, MSSO differs from Web SSO in that it shifts the IdP role from an independent web service to the MNO, thereby further extending the trust delegation chain. MNOs use mobile data sessions (e.g., 4G/5G) to bind network connectivity to subscriber identity, making the network itself a trust anchor and removing the need for third-party interaction. Users only need to tap "verify" once to complete authentication. MSSO has therefore been widely adopted, and all three major Chinese MNOs now support it in a mobile market with over 1.8 billion subscriptions [35].

Originally deployed in thousands of native mobile apps (e.g., TikTok, Alipay) [10, 20, 49], MSSO has been extended to the open web, with major services such as Taobao.com and NetEase Cloud Music adopting Web-based MSSO login flows. In this expansion, providers directly reused native-app SDK architectures on the web without fully considering differences in security properties. In native apps, authentication logic is embedded in proprietary SDKs and compiled into binaries, making the execution environment relatively opaque. However, on the web, the client-side logic runs as JavaScript in the browser. Sensitive data such as credentials, API parameters, and tokens are exposed in readable code, making extraction easier than from compiled binaries. Prior work [49] focuses on native apps, but our study targets the web, where the transparent execution environment changes the threat model. *Web-based MSSO thus lowers the barrier to inspecting and exploiting authentication flows, creating a distinct attack surface.*

2.2 Preliminary Study of Web-based MSSO

Since the deployment architecture and operational workflow of Web-based MSSO[1] remain poorly understood, we first survey mainstream MSSO providers, empirically reconstruct its authentication pipeline, and characterize the MSSO provider landscape, laying the groundwork for the threat model presented in Section 3.

Seed MSSO Provider Collection. To study this architectural shift, we mapped the MSSO service providers and their integration patterns. We searched major engines (e.g., Baidu, Google) and MNO documentation [3, 11–13] using MSSO-related keywords (e.g., "one-tap authentication") derived from the MSSO mechanism's description to identify service providers. Through manual analysis, we confirmed 14 MSSO providers; the detailed list is available in Appendix D of the full version referenced in Open Science. These providers typically expose services via APIs or SDKs. Chinese MNOs dominate the ecosystem, consistent with prior work on native apps [49]. We also identified international MNOs (e.g., Vodafone, Orange, Telefónica, and AT&T) offering similar GSMA Open Gateway/CAMARA Number Verification services [1, 21, 42]. From documentation analysis, we extracted 20 *MSSO Seed Endpoints* (Appendix D), covering key HTTP API patterns and authentication flows.

Authentication Pipeline. Based on these seed endpoints, we further identify websites that enable MSSO services, aiming to analyze real-world deployments and uncover the MSSO authentication workflow. To achieve this, we used these extracted API patterns to perform active pattern matching against the websites on the Secrank Top1M list [47], whose Fully Qualified Domain Name (FQDN)-level popularity ranking is ideal for our goal, since MSSO integration status often varies across different service FQDNs within the same Second-Level Domain (SLD). We confirmed service enablement by actively loading the webpages for these 1M domains, capturing all network traffic, and identifying whether

[1]From this section onward, unless otherwise specified, MSSO refers to Web-based MSSO in the following text.

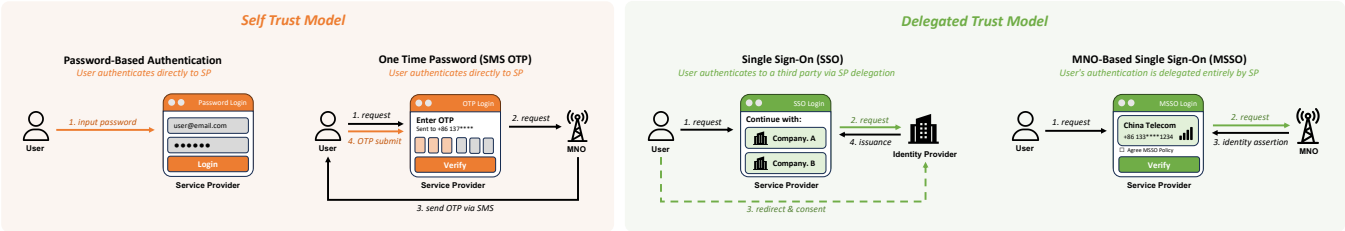


Figure 2: Four Different Types of Authentication

HTTP requests were issued to our *MSSO Seed Endpoints*. Through this process we successfully identified 225 such sites actively utilizing web-based MSSO. For these confirmed sites, we then performed JavaScript call-stack analysis, tracing these requests to their originating scripts. By inspecting the script logic, we derived the parameter handling approach, data-passing mechanisms, and the precise sequence of the real-world authentication flow. Synthesizing this documentation analysis and empirical validation, we characterized the *delegated trust model* in MSSO and generalized the following three-phase workflow for web-based MSSO authentication, as shown in Figure 3. Notably, real-world implementations may vary across different identity providers. For instance, some MNOs employ a two-stage token process where an initial *accessToken* is required to request the final *User Identity Token*; this difference is further analyzed in Section 5.2. Despite minor implementation differences, the core workflow—comprising three phases—remains largely consistent. Our subsequent analysis focuses on these common phases to identify security risks (as discussed in Section 3).

• *Phase 1: Pre-authentication.* Operating over the user's mobile data network, SP's frontend fetches a masked version of user's phone number (e.g., 138xxxx1234) from MNO. This step confirms service availability and prepares the user interface, as shown in Figure 1.

• *Phase 2: Token Issuance.* Following explicit user authorization (e.g., clicking the login button in Figure 1), the SP initiates an MSSO authentication request to the MNO over the user's mobile data network. This request carries the static *developer credentials* (obtained from the MNO when purchasing the service) as verification parameters. The MNO first verifies these *developer credentials* to confirm the SP is authorized to initiate authentication. Once the SP's capability is confirmed, the MNO validates the user's identity based on their active data session. The MNO then encodes the verified identity into a short-lived *User Identity Token* and returns it to the SP's frontend over the same mobile data network.

• *Phase 3: Backend Verification.* In the final phase, the SP's frontend relays the received *User Identity Token* to its own backend. This triggers a server-to-server communication where the SP backend exchanges the *User Identity Token* with the MNO's server to retrieve the user's full phone number. Unlike the first two phases, this server-to-server verification no longer relies on the user's mobile data session, instead completing authentication through a back-end token exchange. The SP backend then establishes a user session and returns the login result to the frontend.

MSSO Service Provider. Building on seed data collection, we further analyze the roles of providers offering MSSO services. In principle, only MNOs possess the inherent capability to perform network-level identity verification based on mobile traffic. However, as developer demand for simplified multi-carrier integration grew, third-party providers entered the workflow as intermediaries, offering this service through a unified interface. Because these two types of providers differ fundamentally in their deployment approaches and trust assumptions, we introduce them in two categories (see Table 1 in Appendix D for the full breakdown).

• *MNOs as Direct Providers.* These are the MNOs (e.g., China Mobile). As the foundational providers, MNOs act as the ultimate identity authority. They exclusively control the network-level mapping between a user's active data session and their verified subscriber identity (i.e., their phone number). They leverage this unique and proprietary capability to offer authentication directly to SPs.

• *Third-Party Resellers.* Internet service providers (e.g., Alibaba Cloud, Netease) lack authoritative network-level identity verification and are therefore not identity sources of truth. Instead, they act as service brokers between MNOs and SPs. As each MNO's proprietary service is incompatible and typically only authenticates its own subscribers, developers face severe fragmentation. To address this fragmentation, resellers purchase authentication capabilities from multiple MNOs and package them as a unified JS SDK or API for developers, abstracting away the fragmented MSSO provider landscape. These unified interfaces detect user's mobile network environment and route requests to corresponding MNO endpoint, allowing one integration to serve users across MNOs.

Despite their different deployment approaches, both provider types share the same client-side delegation pattern: the SP's frontend must embed credentials in JavaScript and follow the three-phase authentication flow. This shared pattern, as we formalize in Section 3, gives rise to a *Trust Hijack Threat*, where delegated trust relationships in MSSO can be hijacked, resulting in unauthorized disclosure of users' verified identities.

3 Trust Hijack Threat in Web-based MSSO

Building on the three-phase MSSO workflow characterized in Section 2.2, we examine trust delegation and validation across the user, SP, and MNO. Using the collected network traffic and JavaScript evidence from confirmed MSSO-enabled websites, we reconstruct the trust chain and pinpoint where its security assumptions break in practice. Based on this analysis, we identify three *Trust Defects* in the delegated trust model that collectively give rise to an overarching *Trust Hijack Threat*, in which an adversary hijacks the trust relationships embedded in the MSSO architecture to silently

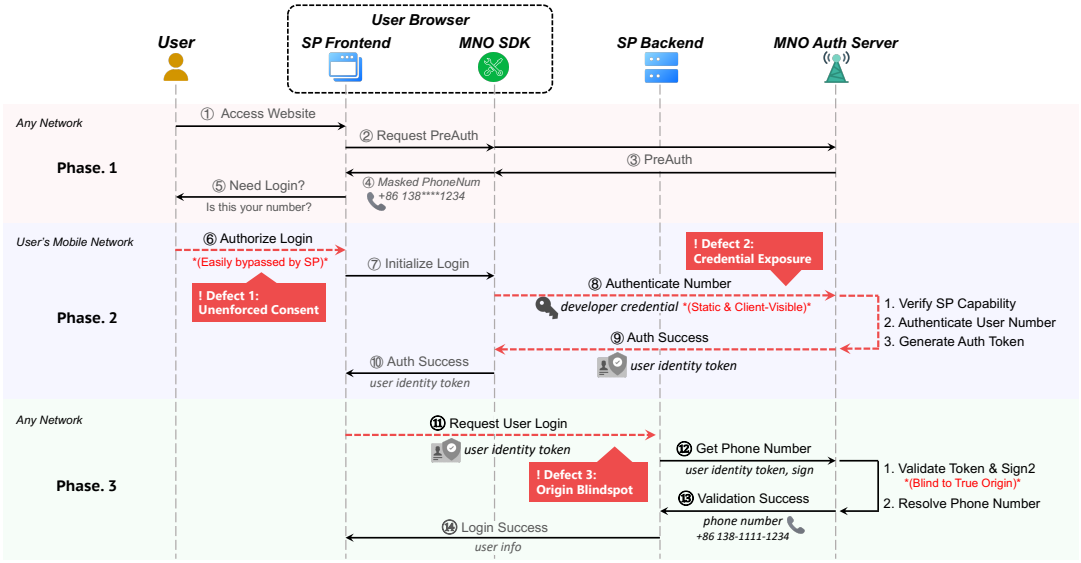


Figure 3: The Three-Phase Workflow of Web-Based MSSO

exfiltrate user identity. This threat manifests concretely as the *One-Click-to-Leak (OCL)* attack, a specific exploitation form where a single webpage visit suffices to leak the user's private identity.

Delegated Trust Defect in MSSO. As detailed in Section 2.2, MSSO relies on a *delegated trust model*. This model transforms the user into a passive authorizer, thereby sidelining them from the core verification loop. The burden of verification thus shifts entirely to the protocol's intermediaries: the SP and the MNO. This shift creates a dissociation of intent and verification and introduces critical vulnerabilities. We identify three defects specific to this *trust model* from our empirical study, as shown in Figure 3.

• *Trust Defect 1: Unenforced User Consent.* In traditional SSO, user consent is securely anchored and verified on IdP's own domain. In contrast, although MSSO nominally includes a UI-based confirmation step, this mechanism lacks verifiable enforcement by the MNO at the protocol level. By allowing an SP with API privileges to use SP-side JavaScript to bypass the interactive UI and directly invoke the authentication API, this architecture violates the federated-identity requirement that identity-attribute release be governed by an authorized-party decision, typically subscriber consent in public-facing transactions [43]. Critically, the MSSO protocol provides no cryptographic or protocol-level binding to prove that a human user actually confirmed the action, meaning the MNO has no reliable way to distinguish a genuine user click from a programmatic API invocation. The MNO blindly trusts the SP's API requests, allowing the authorization flow to be maliciously initiated without the user's actual permission. The same defect exists in native one-tap authentication, where apps may obtain tokens before displaying the SDK's authorization interface [49]. Therefore, this cross-platform defect is not specific to either platform but is inherent in delegating user-consent enforcement to untrusted client-side code.

• *Trust Defect 2: Static Credential Exposure.* To authenticate the SP, the protocol relies on static *developer credentials* (e.g., `appid` and `appkey`), which developers embed in client-side assets such as JavaScript and JSON configuration files. As analyzed in Section 2, the open architecture of the web offers no effective concealment for these embedded credentials. Our analysis finds developer credentials in readable scripts and browser-visible traffic, allowing an adversary to collect reusable SP-authentication material directly from web assets. The same defect exists in native apps, where fixed `appid` and `appkey` values can be recovered from compiled binaries through reverse engineering [49]. Together, these findings show that client-side distribution exposes reusable developer credentials across platforms, while readable web sources lower the extraction barrier.

• *Trust Defect 3: Origin Validation Blindspot.* The MNO lacks contextual awareness to reliably verify the legitimacy of terminal SP and true origin of a request in affected flows. In Reseller-mediated flows, MSSO authorization is mediated by Reseller's *developer credentials* and provider-side routing rather than a verifiable binding between the MNO-facing request and the downstream SP. This creates an origin-validation blindspot: a request carrying valid credentials can appear legitimate even when the actual initiating context differs from that of the registered SP. This blindspot is further compounded by the Third-Party Reseller model, where the MNO observes the trusted Reseller's identity rather than the downstream SP's true origin; we measure the prevalence of this pattern in Section 5.4. The same class of risk could arise in an app deployment if an intermediary authenticates to the MNO on behalf of downstream apps without preserving a verifiable binding to each app, a deployment pattern not reported in prior native one-tap studies [14, 49]. More generally, Trust Defect 3 shows that authenticating an intermediary does not establish the identity of the terminal client unless the delegation is bound end to end. Without such binding identity assertions can escape their intended app, SP, or origin context.

Trust Hijack Threat. Building upon our identification of these Trust Defects, we define the overarching threat enabled by their combination as the *Trust Hijack Threat*. Our threat model targets

MSSO over mobile data. This scope is broad in practice, as mobile data networks provide population-scale coverage and routinely support mobile web access [26, 35]. In this threat, an adversary exploits the trust defects to hijack the delegated trust relationships and execute a hidden private information theft attack that specifically targets the user's phone number and verified identity.

One-Click-to-Leak (OCL) Attack. OCL is a concrete attack of the *Trust Hijack Threat*. In an OCL attack, a user's mere visit without any login action can be sufficient to leak their private identity (e.g., phone number) to a malicious site and even de-anonymize their browsing activity across the web (see Section 6). Specifically, the adversary's strategy is to impersonate a legitimate SP, leveraging the victim's mobile data session to request a *User Identity Token*, and then relay this captured *User Identity Token* to the real SP's legitimate verification endpoint to resolve the victim's private identity. To achieve this goal, the adversary only needs to operate a standard web entry point (e.g., a malicious site or a compromised third-party script) and meet two prerequisites. First, they must obtain the necessary *developer credentials* (e.g., `appid` and `appkey`) of a legitimate SP, a step made possible by *Trust Defect 2* (Static Credential Exposure), which structurally forces these credentials to be statically embedded in transparent web assets. Second, the adversary must be able to exfiltrate the captured *User Identity Token* and use it to resolve the phone number. *Trust Defect 3* (Origin Validation Blindspot) directly enables this, as it permits the *User Identity Token* to be replayed from the adversary's own backend across a different origin and session. Furthermore, *Trust Defect 1* (Unenforced User Consent) compounds the threat by allowing the adversary to trigger the entire authentication flow without the user's actual permission. Notably, an OCL attack does not require the adversary to possess any prior knowledge of the victims' identities or phone numbers. The threat model represents a universal, mass-exploitation strategy. By deploying a script that integrates leaked credentials spanning various MNOs, the adversary can dynamically probe and adapt to the specific carrier of any visiting user. Consequently, the adversary can readily scale this silent identity-stealing attack against a broad population, potentially affecting any user who accesses the entry point via a mobile data session.

To execute this attack, the adversary proceeds through the following four steps.

(1) Luring the Victim: The adversary lures unsuspecting users to a standard webpage (e.g., a seemingly normal site) that secretly hosts the malicious script. The attack triggers as long as the victim accesses this page via a mobile data network.

(2) Covert Token Request: The script silently executes, dynamically adapting to the user's MNO, and uses the corresponding stolen *developer credentials* to trigger a background token request.

(3) Illicit Token Issuance: The MNO, validating only the stolen *developer credentials* and unable to verify the request's origin, cannot distinguish the malicious invocation. It consequently issues a valid *User Identity Token* and delivers it to the adversary.

(4) Attack Completion: Finally, the adversary's backend relays the captured *User Identity Token* to the legitimate verification endpoint (operated by the SP or MNO). This action resolves the *User Identity Token* into the victim's full phone number and finishes the login action, completing the data leakage and even account hijack.

Therefore, a non-powerful adversary who cannot break standard cryptography (e.g., TLS) or compromise the victim's device can execute this attack by exploiting the Trust Defects in MSSO. A single, unsuspecting visit to the adversary's page over a mobile data network can expose the victim's private identity. To confirm the threat's practical feasibility, we conducted controlled experiments using author-controlled SIM cards, mobile-data sessions, and phone numbers. We captured *User Identity Token* values generated through real MSSO flows in these sessions and submitted them from separate sessions to the same SP's legitimate verification endpoint, which returned the corresponding verified phone numbers.

4 OCL Detection

To detect and measure the impact of trust hijack threats in real-world websites, we design and implement *OCL Detector*. It is a hierarchical framework that leverages passive DNS and search engine data to identify MSSO-enabled websites, and further performs active analysis of their authentication behaviors and potential malicious script threats.

4.1 OCL Detector: Challenge, Insight, Overview

Detection Challenge. Detecting and measuring the trust hijack threats at a web-wide scale presents substantial challenges. First, existing malicious website detection methods target generic web-abuse signals [27, 40, 50], whereas OCL-relevant behavior appears in MSSO-specific authentication traffic, including silent API invocations and *User Identity Token* exfiltration. Second, brute-force crawling at web scale is computationally intractable. Third, monitoring real user authentication flows at the network layer would be costly and privacy-invasive.

Detection Insights. Our preliminary study revealed three consistent MSSO deployment properties that guide our detection strategy.

• *Insight 1: MNO as the Authentication Anchor.* The MNO acts as the definitive authentication anchor: every valid MSSO flow must eventually contact an MNO server, either directly or through a Reseller, to resolve the user's identity. This lets our detection focus on MNO interactions and their correlated entities rather than probing the entire web.

•*Insight 2: Stable API Endpoints.* MNO API endpoints exhibit stable, recognizable patterns in their domain names, URL structures, and query parameters, serving as reliable signatures for MSSO traffic. We use the endpoint domains as *MSSO Seed Domains*, enabling a DNS-based correlation that drastically narrows our search scope.

•*Insight 3: Observable Frontend Interaction.* Because MSSO relies on the user's mobile data session for identification, the process is inherently frontend-driven, making the resulting network requests and scripts observable via targeted dynamic analysis.

Overview of OCL Detector. Guided by these insights, we designed and implemented *OCL Detector*, a hierarchical detection framework shown in Figure 4. OCL Detector contains two functional layers. The MSSO identification layer consists of the *MSSO-related Domain Discoverer*, which performs large-scale passive DNS (pDNS) analysis to identify candidate domains correlated with *MSSO Seed Domains*, and the *MSSO Service Verifier*, which translates these domains into candidate URLs and performs targeted dynamic analysis to confirm MSSO integration while collecting evidence. The abuse-analysis

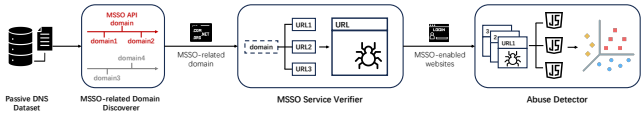


Figure 4: Workflow of *OCL Detector*

layer then uses the collected JavaScript and call-stack evidence to identify suspicious upstream scripts and attribution candidates.

4.2 MSSO-related Domain Discoverer

This module identifies *candidate domains*: domains whose DNS resolutions co-occur with known *MSSO Seed Domains* and are therefore likely associated with MSSO-enabled page loads. We use passive DNS (pDNS) because it records query-response pairs observed at recursive DNS resolvers, providing a proxy for real-world browsing through domain resolution patterns [4, 7, 45]. The analysis proceeds in four steps: constructing per-client query sequences and filtering noisy sources, identifying domains co-occurring with our seed domains, scoring associations with a time-decay model, and aggregating scores across clients with entropy-based weighting.

Query Sequence Construction and Filtering. We construct per-client domain access sequences by grouping DNS queries by source client IP and sorting them by timestamp. Because a single client IP may represent many users behind a large NAT or proxy, its mixed queries can dilute later correlation signals. Following prior pDNS filtering practice [32], we remove clients associated with an abnormally large number of distinct FQDNs per day.

Domain Co-occurrence Analysis. On the filtered sequences, we use a 2.0-second sliding window to capture DNS co-occurrences within the same page-load context. This follows our threat model: a candidate domain and its subsequent *MSSO Seed Domain* call should originate from the same page load, so their DNS resolutions should occur within a brief interval consistent with modern webpage load-time targets such as the 2.5-second LCP threshold in Google's Core Web Vitals [19]. For each query Q_i, we pair it with subsequent queries Q_j where $timestamp(Q_j) - timestamp(Q_i) \leq 2.0$ s. Pairs containing both an *MSSO Seed Domain* and a candidate domain are passed to the scoring model.

Association Scoring Model. For each client IP *ip*, we quantify the association between a candidate domain C and an *MSSO Seed Domain* S via two metrics, adapting a model from prior work [32]. *Support*, $Supp_{ip}(C) = N_{C,ip}/N_{ip}$, measures C's prevalence in the client's query sequence, where $N_{C,ip}$ and N_{ip} denote the number of queries for C and all queries from *ip*, respectively. *Confidence*, $Conf_{ip}(S \Rightarrow C)$, captures how reliably C co-occurs with S, weighted by temporal proximity. For each co-occurrence in a window w, we apply a decay factor $d(S, C, w) = 2^{-\lambda|pos_S(w) - pos_C(w)|}$ that rewards tighter co-occurrences. Summing decay scores across all windows and normalizing by $N_{S,ip}$ yields:

$$Conf_{ip}(S \Rightarrow C) = \frac{\sum_{w \in W_{ip}} d(S, C, w)}{N_{S,ip}}$$

Client-weighted Score Aggregation. A candidate domain appears in sequences from many clients with varying noise levels. We aggregate per-client scores using entropy-based weights: each client's contribution is down-weighted by the Shannon entropy $H(ip)$ of its queried-domain distribution via $w = \frac{1}{1+H(ip)}$, so high-diversity IPs (likely large NATs or proxies) receive lower weight. The final global score for C is the weighted average of all per-client $(Conf_{ip}, Supp_{ip})$ pairs. We additionally discard any candidate resolved by three or fewer client IPs to remove spurious correlations.

The output is a ranked list of candidate domains correlated with *MSSO Seed Domains*, serving as input for the next module.

4.3 MSSO Service Verifier

This module translates candidate domains into concrete URLs and verifies MSSO integration through dynamic probing.

URL Translation and Normalization. We translate candidate domains into URLs using our *website relationship graph*, a large-scale database from a major search engine provider that indexes webpages and models inter-site relationships. Candidate domains fall into two types. For content-providing sites (e.g., `shopping.example.com`), we retrieve URLs directly from the graph. For resource-providing sites (e.g., `sdk.example.js.com`) that only supply embedded JavaScript SDKs or `iframe` content, we use linkage data (`outerchain`, `redirect`, `iframe`) to trace these resources back to the embedding content-providing sites.

The resulting URL list is large and highly redundant; we deduplicate and sample up to 10 representative URLs per FQDN, prioritizing low path similarity, to reduce probing overhead while preserving coverage. This sampling is applied after FQDN-level candidate expansion and prioritizes path diversity; our evaluation below checks its coverage against the independently confirmed Secrank Top 1M ground-truth set.

MSSO Service Verification. We verify MSSO integration by visiting each candidate URL with headless browsers that simulate a mobile network environment and monitoring the resulting traffic. A URL is confirmed when request or response domains and paths match service-signature rules derived from *MSSO Seed Domains* and API patterns (e.g., `/getPreUrl.do`). When a signature is triggered, we record the timestamp, matched rule, matched request/response URL, and the probed root URL in the `security alert log`. The module outputs verified MSSO-integrated URLs, the `security alert log`, and raw evidence for downstream analysis, including HAR files, screenshots, JavaScript resources, and call stack reports. Implementation details of our probing environment are provided in Subsection 4.5.

4.4 Abuse Detector

To understand and trace high-risk activities within the MSSO ecosystem, we design an Abuse Detector. Directly observing user-data exfiltration in transit is impractical because attackers may encrypt or obfuscate stolen identifiers before transmitting them to backends for server-side resolution. Instead of proving final exfiltration on every site, we target the reusable distribution layer: shared upstream scripts that trigger MSSO calls across downstream sites. We perform script analysis via clustering. We encode each script using 24 scalar features across three categories: (1) *General Static Features*, (2) *Provider Features*, and (3) *Specific Behavioral Features*. We further include an averaged path TF–IDF vector and a binary domain multi-hot vector. This representation quantifies the scripts' intrinsic

properties, deployment context, and observed runtime actions. Full feature definitions are linked in the Open Science appendix.

The final clustering result includes both large, well-defined clusters (e.g., benign libraries, official MNO SDKs) and anomalous outliers, providing the candidate set for downstream abuse analysis. We treat the resulting clusters as attribution candidates rather than standalone proof of exfiltration; Sections 6.1 and 6.2 validate them through upstream-provider analysis and case studies.

4.5 Implementation

Dataset. Our study relies on two datasets. (1) *Passive DNS Logs:* We obtained a large-scale Passive DNS (pDNS) dataset from a public DNS service provider in China, covering its recursive resolver traffic over our study period, at around 500 billion DNS requests and 550 million FQDNs per day. (2) *Website Relationship Graph:* We use a large-scale graph from a search engine provider in China that integrates webpage snapshots, DNS records, and search engine logs, including domain-to-domain links, site-to-site resource inclusions, and mappings between URLs, sites, and domains.

Probing Environment and Evidence Collection. Our *MSSO Service Verifier* uses a cluster of `Puppeteer` headless browsers (up to 96 parallel instances via `Puppeteer-Cluster`), each configured with an Android `User-Agent`, mobile screen resolution, and mobile network proxy to satisfy MSSO prerequisites. To capture evidence, we use `Puppeteer-Har` to generate HAR files and implement a *dynamic instrumentation probe* that wraps native browser networking functions (`window.fetch`, `XMLHttpRequest`). When an MSSO API rule is triggered, the probe captures the complete JavaScript call stack via a temporary `Error` object's `.stack` property. To validate this probing environment, we performed controlled configuration checks on MSSO login pages independently confirmed during our preliminary study and spanning MNO-operated and Reseller deployments. We sequentially switched among different MNO data connections and varied four classes of conditions: network routing, client profile, browser state, and automation settings, repeating each condition three times. The results confirmed that our configured environment covered the conditions required to exercise MSSO authentication flows across both deployment types.

Clustering Algorithm. We apply z-score scaling to each of the 24 scalar features across scripts. For each script, we average the L2-normalized TF–IDF vectors of its observed paths and use a binary multi-hot vector for its source domains. We concatenate the three blocks without further scaling. We then apply DBSCAN with Euclidean distance, $\epsilon = 3.2$, and `min_samples`=20. We select ϵ from the transition in the sorted 20-point-neighborhood k-distance curve. The adjacent settings $\epsilon = 3.0/3.2/3.4$ yield 710/586/528 candidates, with noise-set Jaccard similarities of 0.83/1.00/0.90; varying `min_samples` from 10 to 30 retains a Jaccard similarity of at least 0.84. DBSCAN requires no pre-specified number of clusters and identifies low-density samples as anomalous candidates for downstream analysis. We then characterize these candidates through feature analysis and manual review, and identify those associated with abusive MSSO behavior through upstream-provider attribution and case studies.

4.6 Evaluation

We evaluate the *OCL Detector* pipeline from two perspectives corresponding to its key outputs.

Accuracy of MSSO-Enabled Websites. For the websites discovered by the MSSO Service Verifier, we assessed both precision and recall. For precision, we used a two-stage evaluation that separates domain-level authentication evidence from URL-level live authentication. Because URLs under the same apex domain commonly share an MSSO integration, the first stage sampled at the apex-domain level to avoid overweighting sites with many detected pages. We randomly sampled 305 of the 729 detected apex domains (41.8%); the corresponding detections comprised 36,742 URLs. The OCL Detector automatically analyzed the captured traffic for these URLs and produced rule-matched HAR evidence, which we then manually reviewed at the domain level. All 305 sampled domains had reviewed matches containing MSSO authentication evidence. Since live authentication depends on the specific page and path, the second stage randomly sampled 367 of these 36,742 URLs (1.0%) and manually attempted MSSO authentication using author-controlled smartphones over mobile-data connections. Of the 367 URLs, 38 were no longer reachable at validation time. Among the remaining 329 URLs, 319 completed valid MSSO authentication flows (96.9%); three still loaded an MSSO JSSDK but no longer exposed an active authentication flow, while seven fell back to SMS login under their current deployment configuration. For recall, we compared our results against the ground-truth set of MSSO-enabled sites confirmed in the Secrank Top 1M preliminary study (Section 2.2) and found full coverage. These results indicate high precision and high relative recall with respect to our independently confirmed Secrank Top 1M ground-truth set.

Validity of Suspicious Script Clusters. For each cluster, we checked validity through automatic feature analysis and manual review of representative samples. The automated analysis highlights statistical deviations (e.g., content entropy, domain/path patterns, rule triggers), while manual inspection confirms that these signals reflect coherent behaviors rather than noise or coincidental correlations. Detailed analysis is presented in Section 6.

5 Real-World Trust Hijack Assessment

Having formalized the Trust Hijack threat model (Section 3) and built our measurement pipeline (Section 4), we now present our real-world assessment of how widely these Trust Defects manifest in deployed MSSO services. Using the large-scale scans from our OCL Detector, we quantify both the prevalence of MSSO integrations across the web and the extent to which each Trust Defect is exploitable in practice.

5.1 The MSSO Deployment Landscape

Overall Deployment. Using our *OCL Detector*, we processed a pDNS dataset and a large-scale website repository from our search-engine collaborator over a one-year measurement period. Using the *MSSO Service Verifier* (Section 4.3), we identified and confirmed *116,852* distinct URLs across *729* apex domains that actively integrate MSSO services for authentication. To contextualize the real-world defect prevalence, we report statistics at both the URL

and domain levels. URL-level counts reflect the total exposure surface across all observed page instances, while domain-level counts control for site-size bias: a single large site may contribute hundreds of URLs, disproportionately inflating URL-level percentages. The domain-level metric better represents the fraction of affected service providers. Where the two diverge significantly, we discuss the underlying cause.

Evolution of Deployment. Figure 5 plots the number of unique MSSO-enabled domains observed in discrete 10-day windows. We observe a clear overall upward trajectory, confirming the steady adoption of web-based MSSO. This trend empirically validates our premise that MSSO, once primarily confined to native applications, is actively expanding onto the open web platform. This steady growth underscores the increasing importance of the service and the urgency of addressing its underlying security defects. The short-term fluctuations are expected artifacts of our pDNS-based methodology. As pDNS data is a passive sample, it only captures sites actively resolved by users within each specific 10-day window. A dip in the count simply means a subset of sites was not present in the observed DNS records for that period, not that they were decommissioned. The overall trend, which aggregates data over time, provides the key insight into the ecosystem's consistent growth.

Industry Distribution. We used the website classification tool provided by our search-engine collaborator (based on title and body semantics) to categorize these websites and analyze the technology's adoption. We found MSSO deployed across numerous high-value sectors, demonstrating its deep integration into critical web services. Key sectors include:

• *Finance and Insurance* (e.g., `w***bank.com`), where phone numbers are used for high-assurance identity verification.
• *E-commerce and Travel* (e.g., `t***bao.com`, `c***trip.com`), where MSSO reduces friction in checkout and login processes.
• *MNO Self-provisioned Services* (e.g., `1***6.cn`), for customer portal access and managing value-added services.

This widespread adoption demonstrates that the potential attack surface is not niche, but spans critical, high-trust sectors of the web.

5.2 Privileged Tokens Issued Without Consent

Against this broad attack surface, we first examine whether the protocol's user consent mechanism holds in practice. *Trust Defect 1 (Unenforced User Consent)*, defined in Section 3, concerns the MNO's inability to verify that a genuine user action initiated the authentication. As introduced in our workflow analysis (Section 2.2), the MSSO process often begins with a *pre-authentication* (or "pre-fetch") phase. The benign *goal* of this step is to enhance user experience: it fetches a *low-sensitivity* piece of data, typically a masked phone number (e.g., "138****1234"), from the MNO. This mask is then displayed in the UI, allowing the user to confirm which identity they are about to log in with *before* they commit to the action.

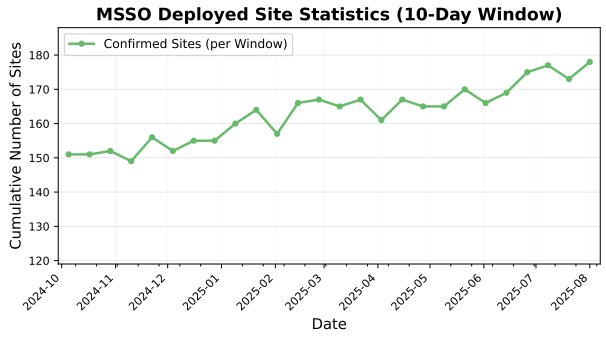


Figure 5: The domain number of unique MSSO-enabled sites observed within discrete 10-day measurement windows.

The core security assumption of this design is that the pre-authentication phase is strictly *low-privilege*. Because it occurs before explicit user consent, it must *only* return non-sensitive data (the mask). The high-privilege token, which can be exchanged for the user's full, verified phone number, must only be generated *after* the user clicks the "Login" button, thereby signaling their intent. However, if this high-privilege token is issued before any explicit user consent, the privilege boundary collapses and the user's verified identifier becomes exposed.

Findings. Our analysis of the dynamic network traffic revealed that this core security boundary is frequently violated. We found that 27.1% (31,654 URLs) of the verified MSSO-integrated pages, corresponding to 24.9% (182 domains), suffer from a critical *consent-bypass flaw*. In these cases, the MNO's API improperly handles the pre-authentication request. The MNO's server prematurely issues a high-privilege token (e.g., an *access_token*) during this initial pre-fetch step. This token, which should have been consent-gated, is delivered to the frontend *before* the user has given any explicit authorization.

Implication. The 27.1% prevalence shows that pre-consent high-privilege token issuance is already widespread under normal browsing, affecting more than one in four verified MSSO-integrated pages. The fact that these high-privilege tokens are issued before explicit user consent demonstrates that the MNO's token-issuance logic does not reliably depend on a verifiable user action. When combined with the exposure of static *developer credentials* measured next, this behavior becomes directly exploitable: an adversary impersonating an SP can deliberately invoke the same pre-fetch API, completely bypass the protocol's user-consent mechanism, and receive the high-privilege token bound to the victim's identity without requiring the user to click any button. As demonstrated earlier by our controlled validation in Section 3, *User Identity Token* values generated without user interaction in the tested flows were successfully submitted from separate sessions to the same SP's legitimate verification endpoint and resolved to the verified phone numbers associated with author-controlled mobile sessions. Together, the large-scale observation establishes the prevalence of pre-consent token issuance, while controlled validation establishes its practical exploitability in the tested flows.

5.3 Leaked and Shared Developer Credentials

The consent-bypass above assumes the adversary can already impersonate a legitimate SP. We now examine the feasibility of this prerequisite. *Trust Defect 2 (Static Credential Exposure)*, defined in Section 3, examines whether the static *developer credentials* that gate SP authentication can withstand exposure in the web's transparent client-side architecture. In the MSSO protocol, this boundary

is enforced by the *developer credentials* (e.g., `appid`, `appkey`) held by the SP. The MNO relies on these *developer credentials* to determine *who* is initiating an authentication request. Our threat model (Section 3) posits that an adversary must first be able to *impersonate* a legitimate, trusted SP. To do this, the adversary must gain possession of these *developer credentials*. The first question of our practical analysis is therefore: *What is the real-world barrier for an adversary to acquire the SP's developer credentials?*

To quantify this risk, we perform this analysis by systematically scanning the `request` and `response` bodies within the HAR files collected by the *MSSO Service Verifier*. We use a set of rule patterns to identify hardcoded MSSO authentication tokens and *developer credentials* (e.g., `appid` and `appkey`). When a match is found, we record a structured entry detailing the website where the leak occurred, the name of the exposed credential, the credential's value, and the MNO we associate with the credential based on its pattern.

Findings. Our analysis reveals that this is a widespread and critical vulnerability. We found that 69.4% (81,146 URLs) of the verified MSSO-integrated pages, affecting 62.6% (457 domains), exposed their *developer credentials*. In total, we identified 460 distinct exposed *developer credentials*. The distribution of these leaks spans all major providers: 270 *developer credentials* belonged to China Mobile (CM), 105 to China Telecom (CT), and 85 to China Unicom (CU). The three MNOs expose developer credentials through structurally distinct channels. CM and CU transmit `appId` values almost exclusively within JSON response bodies (95.4% and 100% of their leaked URLs, respectively), whereas CT exposes `appId` exclusively through URL query strings in 70.4% of its leaked URLs. This distinction carries practical significance: URL-embedded credentials are passively recorded in browser history, proxy logs, and `Referer` headers, making credential recovery possible even without inspecting response bodies, thereby lowering the extraction effort for a potential adversary. This 69.4% exposure rate is consistent with the structural risk identified in Section 3: when reusable *developer credentials* are distributed through browser-visible assets or traffic, the web's open architecture makes their exposure a predictable consequence of this integration pattern rather than the result of isolated developer failures. This widespread exposure can enable reuse of authentication parameters and make malicious requests resemble legitimate SP integrations, weakening the authorization boundary represented by *developer credentials*.

Furthermore, our analysis of the leaked *developer credentials* reveals a high-risk pattern of credential sharing, where a single set of credentials is used by multiple, often seemingly unrelated, websites. This practice severely amplifies the impact of a single leak, as it allows one compromised key to unlock services across many websites. We highlight three representative reuse patterns:

• *Intra-Organizational Reuse:* We found two different *developer credentials* were each hardcoded across 30 distinct websites belonging to the same major e-commerce conglomerate. This pattern indicates that multiple, distinct products from this single entity are all sharing the same credential. It creates a severe amplification of risk: an adversary who obtains this single credential can then impersonate all 30 distinct services, allowing them to leverage one initial compromise across the conglomerate's entire family of products.

• *Cross-Sector Reuse:* Another single credential was shared by 28 different sites, spanning diverse sectors including health, finance, and travel. This pattern suggests these seemingly unrelated companies are all customers of the same third-party Reseller, a clear real-world manifestation of *Trust Defect 3 (Origin Validation Blindspot)* discussed in Section 5.4. The immediate risk is a massive blast radius: one leaked credential lets an adversary impersonate 28 companies through their common insecure intermediary.

• *Widespread Insecure Practices:* We found numerous other instances of sharing, where other *developer credentials* were reused across multiple domains, spanning a wide range of services like insurance and cloud platforms. This demonstrates that such insecure reuse practices are widespread, not isolated to just a few large platforms.

Together, the widespread exposure and cross-site reuse of *developer credentials* demonstrate that the SP–MNO authentication boundary is systematically undermined in practice.

5.4 Origin Blindspot via the Reseller Ecosystem

Even if an MNO attempted to detect impersonation by validating the request's origin, the supply-chain architecture might defeat this defense. We now assess *Trust Defect 3 (Origin Validation Blindspot)*, defined in Section 3, which arises when the Reseller ecosystem obscures the true initiator of an authentication request. To understand the distribution of the supply chain, we systematically analyzed the major providers in our dataset. Appendix D summarizes these key providers, categorizing them by their architectural role (MNO or Reseller) and the nature of their service.

The Reseller model is a prevalent part of the ecosystem. However, this model introduces a distinct origin-validation gap that compounds Defect 3 (the Origin Validation Blindspot). By acting as a trusted intermediary, the Reseller obscures the true identity of the downstream SP initiating the authentication, thereby limiting the MNO's ability to validate the request's true origin.

In the basic MSSO framework, trust is delegated from the user to the SP. This delegation already creates a gap where the MNO cannot be certain of the user's knowing participation. An attacker who steals an SP's *developer credentials* could thus impersonate that SP, request authentication for a victim, and steal their identity. Theoretically, a vigilant MNO might attempt to mitigate this *protocol-level* weakness by enforcing strict source validation, such as checking the request's origin against the SP's registered domain. However, the introduction of the Reseller ecosystem limits this defense in the analyzed flows. As the MNO serves as the ultimate Authentication Anchor, even when a downstream SP (a customer) subscribes to a reseller's service, the authentication process must ultimately be resolved by an MNO-controlled endpoint. The Reseller's SDK mediates this flow, but it culminates in a request to the MNO using the Reseller's identity and *developer credentials*, not the customer SP's. This creates the critical indirection. The MNO's endpoint receives a request that appears valid, as it originates from a trusted partner (the Reseller). However, from the MNO-facing request, the MNO cannot identify the downstream SP. It cannot distinguish a legitimate authentication request from `shopping.example.com` from a malicious one initiated by `attacker-site.com`, as both are masked behind the Reseller's single, valid identity. This dynamic creates the Origin Validation Blindspot defined in Section 3.

To assess the widespread occurrence of this architectural risk, we analyzed the distribution of provider roles. Our analysis reveals

that a significant portion of the ecosystem uses Reseller-mediated integration: ***31.8%*** (232) of the *729* apex domains, corresponding to 19.6% (22,859 URLs), integrate MSSO services via a Third-Party Reseller. This finding demonstrates that the Reseller deployment pattern underlying Defect 3 is not theoretical or niche, but widespread in the wild. In the analyzed flows, request indirection leaves the MNO unable to identify the downstream SP from the MNO-facing request, which can increase the feasibility of the *OCL Threat*.

The Reseller model is associated with a higher observed credential-exposure rate. Reseller-connected URLs exhibit a credential leak rate of 80.1%, compared to 66.6% for sites connecting directly to MNOs (74.2% vs. 57.2% at the domain level). Reseller SDKs support multiple MNOs, and a single page load can expose credentials from multiple MNOs at once. We observed this multi-MNO co-occurrence on 122 domains, with CM and CT credentials appearing together most frequently (115 domains). The same major e-commerce conglomerate whose intra-organizational credential reuse is documented in Section 5.3 exposes credentials spanning all three MNOs across its 30 Reseller-integrated domains, providing a concrete example of multi-MNO co-exposure. Taken together, these measurements show that Reseller-connected deployments have a higher observed exposure rate and place the Reseller rather than the downstream SP at the MNO-facing authentication boundary.

6 One-Click-to-Leak Attack Evaluation

To measure in-the-wild exploitation of the OCL attack, we analyze the attacker supply chain, covering both downstream suspicious scripts and the upstream providers that distribute them. We further reconstruct the abuse and monetization pipeline using sanitized backend code and data, obtained through our industry partner and law enforcement, from one representative seized platform.

6.1 Malicious Provider Discovery

As discussed in Section 4.4, directly observing exfiltration in transit is impractical. We therefore adopt the indirect discovery strategy enabled by our Abuse Detector, combining downstream behavioral analysis to identify suspicious scripts with upstream attribution to trace them to their source providers, revealing the platform-based nature of abusive MSSO exploitation. We first filtered 9,782 unique JavaScript files (from our dynamic probe) for high-risk targets via the Abuse Detector (Section 4.4), with feature engineering (Appendix E) and DBSCAN clustering. This clustering produced 586 anomalous outliers, a baseline Cluster 0 containing 8,729 scripts, and 9 other clusters (benign and suspicious) totaling 467 scripts.

Benign Clusters. Our analysis first successfully grouped the majority of scripts into large, benign clusters with clear, interpretable characteristics. Examples include:

• *Cluster 3 (Official MNO Scripts):* This cluster is characterized by first-party resource loading, high authentication trigger counts, and domains operated by MNOs, identifying these scripts as official operator-provided SDKs and a clear signature of legitimate, high-intent authentication.

• *Cluster 8 (Ad Platforms):* This cluster comprises widely distributed third-party scripts, often with versioned URLs, which manual inspection identifies as advertising and analytics platforms; their near-zero authentication trigger rate shows our features effectively distinguish co-located scripts from those actively involved in authentication.

Suspicious Usage Patterns. Analyzing the smaller suspicious clusters and 586 outliers, we identify several distinct high-risk MSSO integration patterns. In particular, by examining the feature dimensions along which outliers deviate, we group scripts with similar risk profiles, yielding the following categories:

• *Pattern 1: Obfuscated First-Party Logic.* This pattern is defined by scripts loaded as first-party assets, having low prevalence, and exhibiting high content entropy. This profile matches site-specific, custom-developed code that has been intentionally obfuscated. The combination of high relevance to authentication behaviors with obfuscation suggests a deliberate effort by these sites to hide their custom MSSO integration logic, possibly to conceal non-standard data collection or flawed implementations.

• *Pattern 2: Hyper-Active First-Party Integration.* This pattern is also marked by first-party scripts, but its defining characteristic is an extremely high frequency of triggering authentication events. This indicates that MSSO is not just a login option but is a core, high-frequency business function (e.g., triggered on many pages, not just at login). While not inherently malicious, this "hyper-active" pattern creates a significantly larger attack surface and increases risk, as the high-privilege authentication process is invoked far more often than necessary.

• *Pattern 3: Covert Exploit Platform Candidates.* This pattern consists of scripts loaded purely as third-party resources and exhibiting low prevalence. They are not official MNO SDKs and display key high-risk traits: high authentication trigger rate (MSSO as sole purpose) and high content entropy (obfuscation). Together, these traits are consistent with third-party Exploit-as-a-Service platforms that package OCL attacks as unofficial covert solutions for downstream customers.

Finally, this clustering analysis narrowed our investigation from 9,782 scripts to **586 potentially high-risk scripts**, from the suspicious clusters and anomalous outliers. Furthermore, we screened all 586 retained resources. Of the 582 parseable JavaScript resources (the remaining four were JSON-like payloads), we found that 255 (43.8%) contained at least one obfuscation pattern recognized by the pinned AST-based `obfuscation-detector` v3.0.0 [6]. A conservative structural screen further found non-literal `eval` or `Function` construction in 214 resources, explicit debugging-interference structures in 6, and environment-dependent control of a delivery or execution sink in 2. These measurements characterize observable transformations and code structures within the high-risk set.

Upstream Provider Attribution. Having identified 586 potentially high-risk scripts deployed across 564 domains through downstream behavioral clustering, we further trace these scripts upstream to their source providers. We conducted an Upstream Provider Attribution analysis, examining all 729 MSSO-enabled sites identified by our MSSO Service Verifier (Section 4.3). For each site, we extract the source providers of every external subresource and rank these providers by the number of MSSO-enabled sites that reference them, thereby pinpointing the entities most strongly correlated with MSSO deployments. Among the 586 high-risk scripts, 162 (27.6%) were served by 5 upstream platforms, covering 101 websites. The scripts that were the most anomalous were particularly concentrated among just a few of these platforms. This finding strongly

indicates that the observed high-risk MSSO behavior is not the result of sporadic, individual developer error, but rather a platform-based activity distributed through a few upstream providers.

6.2 Case Study: In-the-Wild Exploitation

We now present evidence of how the OCL attack is exploited in practice. We first describe two representative case studies identified through clustering and attribution analysis, which provide strong script- and provider-level association evidence. We then analyze sanitized backend code and data from one representative seized platform; this backend evidence directly confirms one operational OCL exploitation pipeline and reveals its industrialized scale.

Strongly Associated Abuse Patterns. Cross-referencing the downstream behavioral analysis with the upstream source attribution shows that the suspicious patterns are concentrated rather than isolated. Because final user-data exfiltration cannot be reliably observed in transit, we classify these 101 websites as strongly associated with abusive MSSO exploitation through scripts supplied by 5 upstream platforms. Two representative cases illustrate distinct exploitation strategies.

• *Case Study 1: Co-opted Analytics Platform.* This abuse pattern links to Platform A, a high-frequency upstream provider from our attribution. Its clustered anomalous outlier scripts masquerade as user analytics tools but instead secretly trigger the MSSO pre-fetch API (exploiting the Trust Defects characterized in Section 5) on page load, capture the User Identity Token, and exfiltrate it to Platform A's servers, enabling user de-anonymization without consent.

• *Case Study 2: Malicious Service Chaining.* This pattern appears in Platform B's scripts (Cluster 2: Obfuscated First-Party Logic), which chain two Trust Defects: using stolen downstream client *developer credentials* (Trust Defect 2, Section 5.3) for requests, and calling the vulnerable pre-fetch API (Trust Defect 1, Section 5.2) to obtain high-privilege User Identity Tokens without user interaction. These tokens are instantly exfiltrated to a third-party data-tracking platform, completing the malicious service chain.

The OCL Black-Hat Economy. Our case studies establish strong associations between abusive MSSO behavior and identifiable upstream platforms. To understand the resulting economy, we reconstruct one representative seized platform's operational and financial pipeline from exploit delivery to monetization. Specifically, through our collaborator, a leading search engine company, we traced suspicious scripts to upstream platforms using our OCL Detector pipeline (Section 6.1) and reported the findings to law enforcement. The subsequent investigation and takedown operation resulted in the seizure of several such platforms. Our analysis uses sanitized backend code and data from one representative seized platform, with all personally identifiable information removed prior to analysis.

This seized platform operates as a multi-actor criminal enterprise, with roles mirroring legitimate business structures. To clarify these roles within the black-hat economy, we define the following: 1) Victim is the end-user browsing the web and using search engines over a mobile data network; 2) Platform is the black-hat platform providing the exploit as a service; 3) Customer is the malicious website operator who pays the *Platform* to deploy the exploit on their site; 4) Affiliate is the customer who is also incentivized to resell the Platform's user de-anonymization service to other Customers.

Our analysis reveals the Platform's three-stage industrial architecture. The *upstream* is a sophisticated Software-as-a-Service (SaaS) platform offering OCL exploits as a product, distributed via a multi-level Affiliate network with industrialized rules (e.g., 1.5 CNY minimum per stolen number, minimum purchase quotas). The *midstream* is an automated two-stage data-fusion pipeline: first capturing Victims' basic identifiers (phone number, IP, referrer), then enriching profiles by leveraging Customers' webmaster/analytics credentials to pull and associate Victims' search keywords. The *downstream* handles monetization via anonymous USDT payments and a fully automated commission system incentivizing Affiliates.

Upstream: The SaaS Exploit-Delivery Platform. The Platform's upstream layer functions as a centralized SaaS provider that productizes exploit capabilities, with an admin panel defining business rules for Affiliates who resell de-anonymization services. Backend scripts enforce this hierarchy by validating credit quotas and price controls when provisioning Customers. When Victims visit Customer sites, malicious scripts identify MNOs via IP geolocation and deploy tailored exploits to extract phone numbers or user identity tokens, while remaining highly resistant to detection:

• *Cloaking and Authentication:* The script is disguised as a common CDN library. The delivery URL uses unique subdomains and path components (e.g., `dz37cb.malsite.com/zfww/jqurey.js`). Backend logic confirms that these URL components function as a method to authenticate paid Customers before serving malicious payload.

• *Anti-Analysis:* The scripts are replete with anti-analysis checks, such as checking that the request includes a mobile `User-Agent` and has a valid `HTTP_REFERER`. Furthermore, we observed it employs techniques to actively thwart debugging, such as disabling all `console` functions on the Victim's webpage.

• *Stealthy Exfiltration:* The client-side payload, dynamically generated by the server, uses a stealthy exfiltration chain. It dynamically creates a hidden `img` tag and sets the MNO's authentication URL as its `src` attribute. The image `onload` event is used as a callback to trigger an `XMLHttpRequest`, which stealthily sends the captured token or phone number back to the Platform's backend.

Midstream: Automated Victim Profile Enrichment. The Platform's primary value is not just collecting phone numbers, but enriching them, with a two-stage automated pipeline.

• *Stage 1: Initial Capture.* The MNO-specific exploit scripts perform the first level of enrichment. Upon successfully stealing the phone number, they execute a `SQL INSERT` query to write the Victim's profile into the main `data` table. It binds the phone number to the Victim's `referrer` (the webpage they came from), `IP`, IP-based geolocation, User-Agent, and the Platform's Customer ID (`username`).

• *Stage 2: Server-Side Keyword Fusion.* The Platform enriches this data with Victims' search keywords via an automated backend task. It queries the **user** table for Customers who provided webmaster analytics credentials (e.g., usernames, passwords, access tokens), then uses these to periodically fetch visitor logs (IP and search keywords) from the search engine's analytics service. The backend auto-fuses this data, matching keywords to Victims' profiles via IP to link their phone numbers with high-intent search queries.

Downstream: Automated Monetization and Commission. With enriched victim profiles in hand, the Platform turns to monetization. To evade financial regulation, it uses cryptocurrency: its Customer-facing payment page directs Affiliates to pay via USDT (Tether)

on the TRC20 network, with a static wallet address. Payment is fully automated—a backend script leverages a crypto exchange API to monitor deposits and auto-credit Customer accounts. Beyond payment processing, the Platform incentivizes distribution through an automated, multi-level-marketing (MLM) commission structure. Our analysis of the backend code shows that when a Customer makes a payment, the script traverses up the Affiliate chain. It retrieves the pre-set cost basis for both the Affiliate and their parent Affiliate, calculates the profit margin on the sale, and immediately credits the parent Affiliate's account with their commission, converted back into service credits. This automated, pyramid-style commission structure creates a powerful financial incentive for Affiliates to perpetuate and resell the OCL exploit.

The Platform's database shows that it covertly obtained and profiled *14,100 unique Victim phone numbers in just three days*. Based on the Platform's minimum price floor of 1.5 CNY per number, this three-day collection period represents potential illicit revenue of over 21,000 CNY (approx. $2,900 USD). This confirms that OCL attacks are active, scalable, and highly monetized in the wild.

7 Discussion

Recommendation. To enhance the security of web-based MSSO and protect user privacy, we propose the following mitigations:

• *Gate Token Issuance with User Verification.* MNOs should gate *User Identity Token* issuance on a low-friction user-verification step, rather than relying only on a consent click after displaying a masked phone number. The key requirement is that the user provide information not exposed by the conventional one-tap flow, preventing an automated script from completing authentication alone. One implementation is *Masked-Number Completion Verification* (Appendix F): the interface leaves the middle digits in the masked number blank (e.g., `138xxxx1234`) and requires the user to enter those digits before token issuance. This mechanism adds explicit user participation while preserving MSSO's low-friction experience; MNOs should pair it with server-side rate limiting or temporary blocking to prevent brute-force attempts.

• *Enforce Strict Privilege Separation.* To mitigate the consent-bypass vulnerability (27.1% of verified MSSO-integrated URLs), MNOs should strictly prevent pre-authentication endpoints from issuing high-privilege tokens and ensure token generation occurs only after explicit user consent via a dedicated authentication endpoint.

• *Deprecate Static Developer Credentials.* The widespread leakage of *developer credentials* (in 69.4% of verified MSSO-integrated URLs) demonstrates their insecurity. MNOs are encouraged to deprecate static `appid`/`appkey` mechanisms in favor of a more secure model, such as short-lived dynamic tokens fetched by the SP backend.

Responsible Threat Disclosure. We provide several repair recommendations and have responsibly disclosed our findings to affected MNOs, SPs, and relevant stakeholders, who have acknowledged this threat and are positively working on remediation. Encouragingly, following our disclosure, we observed that at least 148 affected sites among the domains identified by OCL Detector had deployed recommended defenses, including safer MSSO implementations such as masked-number completion. This observation demonstrates a practical way to reduce OCL exposure without abandoning MSSO's low-friction login experience.

Limitation. Despite significant efforts in understanding web-based MSSO, our work still has limitations. First, the datasets we used (PDNS and search engine index data) are sourced from Chinese vendors, introducing geographical limitations to our evaluation results. However, both datasets serve a large user base: the PDNS dataset includes around 500 billion unique DNS requests for about 550 million FQDNs daily, and the search engine data covers over 20 million users with hundreds of millions of indexed pages. Thus, although deployment prevalence and provider practices may differ across regions, our large-scale datasets still reveal the deployment status and real-world risks of a mature web-based MSSO ecosystem, and these findings can generally inform similar MNO-based authentication deployments globally. Second, our method for identifying MSSO-enabled websites may have limitations. On one hand, data noise may cause bias when correlating PDNS with candidate MSSO-related domains. But, we verified this step using actively collected MSSO Seed Endpoints (Section 2.2), with no false positives detected. Websites may also condition MSSO delivery on the client or network environment, which can cause false negatives during dynamic probing. To reduce this risk, we conducted the controlled validation described above and configured the probing environment to preserve the conditions required for ordinary MSSO authentication flows while avoiding unnecessary changes to the browser environment. Nevertheless, adaptive cloaking can withhold a payload before collection, while obfuscation and environment-dependent branches can restrict source and runtime analysis to delivered and executed paths. Our traffic signatures and call-stack capture reduce reliance on readable source code, but cannot recover withheld or unexecuted behavior. The detected deployment and abuse counts are therefore conservative lower bounds under such evasion, while the positive responses to our disclosures confirm the risk's widespread impact. On the other hand, the clustering-based abuse detector leveraging static and dynamic page features may produce biased results. However, we manually validated randomly sampled results from each cluster, finding no ambiguity. Together, the controlled validation and manual review support the observed positive results, while residual evasion primarily risks false negatives.

8 Related Works

Mobile Authentication Security. Prior research has characterized security limitations across the authentication mechanisms that precede MSSO. Studies of passwords assess the security and usability limitations of user-managed secrets [8, 9], while studies of MNO-assisted authentication identify SMS OTP interception on mobile devices and weaknesses in MNO authentication procedures for SIM swaps [29, 30]. As authentication shifts toward Web SSO, research characterizes IdP–SP architectures and security properties, including formal analyses of OAuth/OIDC protocols [2, 15, 34]. Empirical and automated studies examine security flaws in deployed SP integrations and provider libraries [38, 41, 44, 48], while other work investigates account and session compromise and automatic privacy leakage during visits to SPs [17, 46]. These studies focus on Web IdP accounts, protocol tokens, SP integrations, and web sessions, whereas web-based MSSO shifts the identity authority to the MNO, which maps an active mobile-data session to a verified phone identity. Recent MSSO work identifies vulnerabilities

in cellular-network-based one-tap authentication, including unverifiable network authenticity [14] and the MNO server's inability to distinguish which app initiated authentication, leading to the *SIMulation attack* [49]. In contrast, we study web-based MSSO at scale and show how leaked developer credentials, consent bypass, and weak origin validation enable OCL.

Phone Numbers as an Abuse Primitive. Prior work has shown that phone numbers are high-value identifiers for fraud, profiling, and targeted abuse. Studies of telephony fraud and mobile telephony threats show how attackers monetize the reachability and trust associated with phone numbers [5, 39], while work on cross-application features and contact discovery shows how phone numbers enable targeted attacks and large-scale user enumeration [22, 23]. Web phishing represents another major path for identity compromise, relying on social engineering and deceptive interfaces to harvest credentials [25, 31]. OCL occupies a different point in this attack space: an adversary can operate an ordinary web entry point, reuse exposed developer credentials, and turn a standard mobile-data page visit into leakage of a verified mobile identity. This low-barrier, high-value property makes OCL attractive for monetized abuse, because leaked phone numbers can be immediately linked to web context such as referrers, IPs, and search intent.

9 Conclusion

MSSO is an emerging password-free authentication framework leveraging users' active mobile data sessions. Unlike traditional SSO, it shifts the IdP role from web services to MNOs and delegates the authentication anchor to SPs. Through empirical analysis of MSSO mechanisms, we identify three Trust Defects that collectively give rise to a Trust Hijack Threat; OCL is the concrete attack form that stealthily steals login-sensitive data. Collaborating with a leading search engine company, we designed a three-stage detection framework for the first large-scale assessment of the MSSO ecosystem, spanning 116,852 URLs across 729 apex domains and quantifying its prevalence, key providers, and observed Trust Defect manifestations (Trust Defect 1: 27.1% of URLs exhibit pre-consent high-privilege token issuance; Trust Defect 2: 69.4% of URLs expose developer credentials; Trust Defect 3: 31.8% of apex domains use Reseller-mediated integration, the deployment pattern underlying this defect). We present the first real-world evidence of OCL attacks: our analysis identifies 101 websites strongly associated with OCL attack behavior through scripts served by 5 upstream platforms. Sanitized backend evidence from one representative seized platform also reveals an operational OCL exploitation pipeline that collected 14,100 unique phone numbers within three days and monetized the resulting data. We propose mitigations and have conducted responsible disclosure; together, these efforts help improve mobile authentication security.

References

[1] Aduna. 2025. AT&T, T-Mobile and Verizon Come Together to Bring First Standardized 5G Network APIs to the U.S. Leveraging Aduna. https://www.adunaglobal.com/media/a2npz04v/att-t-mobile-verizon-5g-network-apis.pdf

[2] Furkan Alaca and Paul C. van Oorschot. 2020. Comparative Analysis and Framework Evaluating Web Single Sign-on Systems. *ACM Comput. Surv.* 53, 5 (2020), 112:1–112:34. doi:10.1145/3409452

[3] Alibaba Cloud. 2026. Phone Number Verification Service: Developer Reference. https://help.aliyun.com/zh/pnvs/developer-reference/

[4] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, Tadayoshi Kohno (Ed.). USENIX Association, 491–506.

[5] Marco Balduzzi, Payas Gupta, Lion Gu, Debin Gao, and Mustaque Ahamad. 2016. MobiPot: Understanding Mobile Telephony Threats with Honeycards. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang (Eds.). ACM, 723–734. doi:10.1145/2897845.2897890

[6] Ben Baryo. 2026. obfuscation-detector. Software, version 3.0.0. https://github.com/ctrl-escp/obfuscation-detector/releases/tag/v3.0.0

[7] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society.

[8] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*. IEEE Computer Society, 553–567. doi:10.1109/SP.2012.44

[9] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2015. Passwords and the evolution of imperfect authentication. *Commun. ACM* 58, 7 (2015), 78–87. doi:10.1145/2699390

[10] China Mobile. 2018. Innovative Practice of Mobile Authentication. https://www.gsma.com/identity/wp-content/uploads/2018/07/MWCS18-Innovative-Practice-of-Mobile-Authentication-Huang-Xi-China-Mobile.pdf

[11] China Mobile. 2026. Internet Capability Open Platform: One-Click Login SDK Developer Guide. https://dev.10086.cn/

[12] China Telecom. 2026. Tianyi Account Open Platform: One-Click Login SDK Developer Guide. https://id.189.cn/

[13] China Unicom. 2026. China Unicom Online: One-Click Login Authentication. https://www.onlineinfotech.cn/product/detail?productId=39b6528d-0d1d-44

[14] Zhiwei Cui, Baojiang Cui, Junsong Fu, and Bharat K. Bhargava. 2023. An Attack to One-Tap Authentication Services in Cellular Networks. *IEEE Trans. Inf. Forensics Secur.* 18 (2023), 5082–5095. doi:10.1109/TIFS.2023.3304840

[15] Daniel Fett, Ralf Küsters, and Guido Schmitz. 2016. A Comprehensive Formal Security Analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1204–1215. doi:10.1145/2976749.2978385

[16] United States. National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. 1978. *The Belmont report: ethical principles and guidelines for the protection of human subjects of research*. Department of Health, Education and Welfare.

[17] Mohammad Ghasemisharif, Amrutha Ramesh, Stephen Checkoway, Chris Kanich, and Jason Polakis. 2018. O Single Sign-Off, Where Art Thou? An Empirical Analysis of Single Sign-On Account Hijacking and Session Management on the Web. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15–17, 2018*. USENIX Association, 1475–1492.

[18] Google. 2025. Sign in with Google for Web — Overview. https://developers.google.com/identity/gsi/web/guides/overview

[19] Google. 2025. Understanding Core Web Vitals and Google Search Results. https://developers.google.com/search/docs/appearance/core-web-vitals

[20] GSMA. 2017. Mobile Authentication: Capitalising on China's Identity Market. https://www.gsma.com/solutions-and-impact/technologies/mobile-identity/wp-content/uploads/2017/12/MC-China-Mobile-report-English-FINAL.pdf

[21] GSMA. 2024. Open Gateway NBI APIs Realisation in the SBI. https://www.gsma.com/solutions-and-impact/technologies/networks/wp-content/uploads/2024/02/OPG.09-V1.0-NBI_SBI-Realisation.pdf

[22] Srishti Gupta, Payas Gupta, Mustaque Ahamad, and Ponnurangam Kumaraguru. 2016. Exploiting Phone Numbers and Cross-Application Features in Targeted Mobile Attacks. In *Proceedings of the 2016 ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM@CCS 2016, Vienna, Austria, October 24-28, 2016*. ACM, 73–82. doi:10.1145/2994459.2994463

[23] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, and Thomas Schneider. 2021. All the Numbers are US: Large-scale Abuse of Contact Discovery in Mobile Messengers. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, Virtual Event, February 21-25, 2021*. The Internet Society. doi:10.14722/ndss.2021.23159

[24] Dick Hardt. 2012. The OAuth 2.0 Authorization Framework. RFC 6749. doi:10.17487/RFC6749

[25] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David A. Wagner. 2019. Detecting and Characterizing Lateral Phishing at Scale. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*. USENIX Association, 1273–1290.

[26] International Telecommunication Union. 2025. Measuring Digital Development: Facts and Figures 2025 – Mobile Network Coverage. https://www.itu.int/itu-d/reports/statistics/2025/10/15/ff25-mobile-network-coverage/
[27] John McGahagan IV, Darshan Bhansali, Ciro Pinto-Coelho, and Michel Cukier. 2019. A Comprehensive Evaluation of Webpage Content Features for Detecting Malicious Websites. In *9th Latin-American Symposium on Dependable Computing, LADC 2019, Natal, Brazil, November 19-21, 2019*. IEEE, 1–10. doi:10.1109/LADC48089.2019.8995713
[28] Erin Kenneally and David Dittrich. 2012. The Menlo Report: Ethical principles guiding information and communication technology research. SSRN 2445102.
[29] Kevin Lee, Benjamin Kaiser, Jonathan R. Mayer, and Arvind Narayanan. 2020. An Empirical Study of Wireless Carrier Authentication for SIM Swaps. In *Sixteenth Symposium on Usable Privacy and Security, SOUPS 2020, August 7-11, 2020*, Heather Richter Lipford and Sonia Chiasson (Eds.). USENIX Association, 61–79.
[30] Zeyu Lei, Yuhong Nan, Yanick Fratantonio, and Antonio Bianchi. 2021. On the Insecurity of SMS One-Time Password Messages against Local Attackers in Modern Mobile Devices. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.
[31] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. USENIX Association, 3793–3810.
[32] Baojun Liu, Zhou Li, Peiyuan Zong, Chaoyi Lu, Hai-Xin Duan, Ying Liu, Sumayah A. Alrwais, XiaoFeng Wang, Shuang Hao, Yaoqi Jia, Yiming Zhang, Kai Chen, and Zaifeng Zhang. 2019. TraffickStop: Detecting and Measuring Illicit Traffic Monetization Through Large-Scale DNS Analysis. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 560–575. doi:10.1109/EUROSP.2019.00047
[33] Siqi Ma, Runhan Feng, Juanru Li, Yang Liu, Surya Nepal, Diethelm Ostry, Elisa Bertino, Robert H. Deng, Zhuo Ma, and Sanjay Jha. 2019. An empirical study of SMS one-time password authentication in Android apps. In *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, David M. Balenson (Ed.). ACM, 339–354. doi:10.1145/3359789.3359828
[34] Christian Mainka, Vladislav Mladenov, Jörg Schwenk, and Tobias Wich. 2017. SoK: Single Sign-On Security - An Evaluation of OpenID Connect. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 251–266. doi:10.1109/EUROSP.2017.32
[35] Ministry of Industry and Information Technology of the People's Republic of China. 2026. 2025 Statistical Bulletin of the Communications Industry. https://www.miit.gov.cn/jgsj/yxj/xxfb/art/2026/art_bea806f4dd20457cb0158795cc210aa7.html
[36] David M'Raïhi, Mihir Bellare, Frank Hoornaert, David Naccache, and Ohad Ranen. 2005. HOTP: An HMAC-Based One-Time Password Algorithm. *RFC* 4226 (2005). doi:10.17487/RFC4226
[37] David M'Raïhi, Salah Machani, Mingliang Pei, and Johan Rydell. 2011. TOTP: Time-Based One-Time Password Algorithm. RFC 6238. doi:10.17487/RFC6238
[38] Tamjid Al Rahat, Yu Feng, and Yuan Tian. 2022. Cerberus: Query-driven Scalable Vulnerability Detection in OAuth Service Provider Implementations. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, 2459–2473. doi:10.1145/3548606.3559381
[39] Merve Sahin, Aurélien Francillon, Payas Gupta, and Mustaque Ahamad. 2017. SoK: Fraud in Telephony Networks. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 235–250. doi:10.1109/EuroSP.2017.40
[40] Kyle Soska and Nicolas Christin. 2014. Automatically Detecting Vulnerable Websites Before They Turn Malicious. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 625–640.
[41] San-Tsai Sun and Konstantin Beznosov. 2012. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 378–390. doi:10.1145/2382196.2382238
[42] Telefónica. 2024. Spanish Mobile Industry to Launch Online Anti-Fraud and Identity Services through GSMA Open Gateway Initiative. https://www.telefonica.com/en/communication-room/press-room/spanish-mobile-industry-to-launch-online-anti-fraud-and-identity-services-through-gsma-open-gateway-initiative/
[43] David Temoshok, Justin P. Richer, Yee-Yin Choong, James L. Fenton, Naomi Lefkovitz, Andrew Regenscheid, and Ryan Galluzzo. 2025. Digital Identity Guidelines: Federation and Assertions. NIST Special Publication 800-63C-4. doi:10.6028/NIST.SP.800-63C-4
[44] Rui Wang, Shuo Chen, and XiaoFeng Wang. 2012. Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 365–379. doi:10.1109/SP.2012.30
[45] Florian Weimer. 2005. Passive DNS Replication. 17th Annual FIRST Conference on Computer Security Incident Handling (FIRST '05), Singapore.
[46] Maximilian Westers, Andreas Mayer, and Louis Jannett. 2024. Single Sign-On Privacy: We Still Know What You Did Last Summer. In *Annual Computer Security Applications Conference, ACSAC 2024, Honolulu, HI, USA, December 9-13, 2024*. IEEE, 321–335. doi:10.1109/ACSAC63791.2024.00039
[47] Qinge Xie, Shujun Tang, Xiaofeng Zheng, Qingran Lin, Baojun Liu, Haixin Duan, and Frank Li. 2022. Building an Open, Robust, and Stable Voting-Based Domain Top List. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 625–642. https://www.usenix.org/conference/usenixsecurity22/presentation/xie
[48] Yuchen Zhou and David Evans. 2014. SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, 495–510.
[49] Ziyi Zhou, Xing Han, Zeyuan Chen, Yuhong Nan, Juanru Li, and Dawu Gu. 2022. SIMulation: Demystifying (Insecure) Cellular Network based One-Tap Authentication Services. In *52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2022, Baltimore, MD, USA, June 27-30, 2022*. IEEE, 534–546. doi:10.1109/DSN53405.2022.00059
[50] Rasha Zieni, Luisa Massari, and Maria Carla Calzarossa. 2023. Phishing or Not Phishing? A Survey on the Detection of Phishing Websites. *IEEE Access* 11 (2023), 18499–18519. doi:10.1109/ACCESS.2023.3247135

A Open Science

The full version of this paper, including the complete Open Science and Ethical Considerations statements, an appendix with detailed provider and feature tables, and supplementary workflow figures, is available at https://arxiv.org/abs/2609.12037. Our artifact at https://doi.org/10.5281/zenodo.22733454 provides *OCL Detector* source code and seed-domain rules, a sanitized notebook, aggregate DBSCAN outputs, remediation examples, and a controlled OCL demonstration using only author-controlled SIM cards, mobile-data sessions, and phone numbers. Partner agreements prohibit redistribution of pDNS and search-engine data; seized-platform data is restricted to anonymized internal analysis. To protect affected services and prevent abuse, we withhold raw measurements, HAR logs, developer credentials, site identities, script-level records, and attack code. These restrictions permit inspection of the implementation and selected aggregate analyses but prevent full reproduction of our measurements.

B Ethical Considerations

Following the Belmont and Menlo Reports [16, 28], all experiments and data handling were conducted under our industry collaborator's legal and compliance oversight. Probing and token-to-phone validation used only author-controlled SIM cards, mobile-data sessions, and phone numbers; we neither redeemed third-party users' User Identity Tokens nor attempted to obtain their phone numbers. Probing covered only publicly indexed pages, with at most one request per second per target and a server-hosted study notice, contact email, and opt-out mechanism. We handled pDNS and search-engine data under partner agreements, accessing only collaborator-hashed pDNS client identifiers and sanitized or aggregated seized-backend evidence, including salted phone-number hashes, without raw sensitive records. Through our collaborator, we disclosed risks to affected MSSO Providers and platforms and reported affected pages to the national CERT. Our collaborator reported suspected abuse to national law enforcement; the subsequent investigation and takedown supplied the sanitized evidence in Section 6.2.