JSSSEE
Journal of Sustainable Smart Systems in Education & Environment

Journal of Sustainable Smart Systems in Education & Environment (JSSSEE) | eISSN: 2979-143X

Harbor: Phishing- and AiTM-Resistant Authentication with Recoverable Origin-Bound Passwords and Zero Server Changes

Malek Saleh Al-jbour, Department of Computer Science, New Mexico State University, Las Cruces, USA, mjbour@nmsu.edu, ORCID: [0009-0005-1870-5505]

Donovan Fuqua, College of Business, New Mexico State University, Las Cruces, USA, dfuqua@nmsu.edu, ORCID: [0000-0002-6163-1741]

Corresponding Author: Malek Al-Jbour, Mjbour@nmsu.edu

JSSSEE | eISSN: 2979-143X | Vol. 1 No. 2 (2026) | OPEN ACCESS
https://doi.org/10.66823/81s9pf94

Abstract

Origin-bound password derivation makes a phished credential useless anywhere except the site on which it was captured, and it can be deployed with no server-side changes. The most complete prior client-only realization, AnchorPass (Al-jbour, 2026), nevertheless carries three practical costs that block adoption: a device-held secret (pepper) whose loss silently invalidates every account with no recovery path; a memory-hard hash recomputed on every login; and full-origin binding that breaks legitimate multi-subdomain sites. We present Harbor, a Chromium-based browser prototype whose trusted chrome contains a native credential bar below the address bar, and ROBD (Recoverable Origin-Bound Derivation), a scheme that resolves all three costs without weakening the security guarantee. ROBD replaces the pepper with a 128-bit recovery seed rendered once as a checksummed recovery code, preserving the same $2^{H(P)+128}$ offline hardness while making every site password recoverable on any device. It amortizes memory-hardness by running Argon2id once per unlock to produce a root key, then deriving per-site tokens with HKDF in under a millisecond. It scopes tokens to the registrable domain (eTLD+1) with public-suffix–aware separation of hosted-subdomain platforms, and separates HTTP from HTTPS. Because the credential bar lives in browser chrome, the master password is never typed into a page, and a credential firewall blocks the case where a user does so anyway. Across measured experiments—1,000 domain-binding trials, 1,005 cross-domain replay and adversary-in-the-middle (AiTM) relay attempts spanning 15 phishing techniques, latency benchmarks, an offline-cost analysis, a 12-pattern login-UI compatibility suite, and a 200-trial recovery test—Harbor achieves 0.0% replay success, 0.05 ms mean per-login derivation (vs. approximately 118 ms for the prior work), exact recovery in 200/200 trials, and correct field handling on 12/12 UI patterns. Harbor attains passkey-level phishing resistance while retaining password-level deployability and eliminating the loss and performance penalties of prior client-only schemes.

Keywords: Phishing, Browser Security, Origin Binding, Password Derivation, Argon2id, Key Recovery, Usable Security

1. Introduction

Phishing remains the leading route to account compromise, and user education has a well-documented ceiling: in controlled studies, even security-aware participants disclose secrets to convincing look-alike pages 30–40% of the time (Verizon, 2024; Dhamija et al., 2006; Alsharnouby et al., 2015). Rather than trying to drive that residual error rate to zero, we adopt a different design stance: the error itself should carry no cost. A secret entered on the wrong page should be worthless to whoever receives it. Origin-bound passwords realize this stance by letting the browser transform the user's secret into a token cryptographically tied to the active origin; a token phished at D' does not authenticate at D, and the server verifies it exactly as it would an ordinary password (Ross et al., 2005).

Passkeys/WebAuthn and OPAQUE deliver first-class, phishing-resistant authentication but require server integration and coordinated enrollment (FIDO Alliance, 2025; Jarecki et al., 2018; Bonneau et al., 2012). Client-only origin binding needs no such coordination, but the most complete prior realization, AnchorPass (Al-jbour, 2026), inherits three limitations that we argue are the real barriers to deployment:

1. **Pepper fragility.** AnchorPass mixes a 128-bit device-held pepper into each derivation. This is excellent for offline hardness, but the paper's own ethics section concedes that losing the device (or resetting the pepper) invalidates every site token with no recovery path. For a primary authentication mechanism this is a catastrophic, silent failure mode.

2. **Per-login memory-hardness.** Argon2id (128 MB) runs on every login, costing approximately 118 ms. This is tolerable but unnecessary: the memory-hard step defends against an offline attacker and need not gate every interactive login.

3. **Full-origin binding.** Binding to the full origin makes accounts.google.com and mail.google.com derive different passwords, breaking single sign-on and multi-subdomain sites—a correctness problem that pushes users to disable protection.

We introduce Harbor, a Chromium-based browser prototype that moves origin binding from an extension popup into the browser's own trusted UI, and ROBD, a derivation scheme that resolves all three limitations without weakening the security guarantee.

Contributions. (i) ***Recoverable derivation****:* a 128-bit recovery seed replaces the non-recoverable pepper; it is shown once as a checksummed, human-typable recovery code and sealed at rest under both a PIN and the OS keychain; offline

hardness is unchanged at $\geq 2^{H(P)+128}$, and device loss is fully recoverable (200/200 exact restores).

(ii) ***Amortized memory-hardness****:* Argon2id runs once per session unlock to produce a root key RK; per-site tokens are HKDF-SHA256 expansions (Krawczyk & Eronen, 2010); measured per-login latency drops from approximately 118 ms to 0.05 ms with no reduction in offline attacker cost.

(iii) ***Registrable-domain scoping****:* tokens bind to eTLD+1 via the Public Suffix List (Mozilla Foundation, n.d.), with public-suffix–aware separation of hosted-subdomain platforms (*.github.io) and scheme separation (HTTP vs. HTTPS).

(iv) ***Browser-native credential bar and firewall****:* a trusted credential bar in browser chrome removes AnchorPass's manual-invocation recall burden and its spoofable-popup surface; a credential firewall blocks submissions that carry the raw master password.

(v) ***Universal login-UI handling and rotation****:* a detector covering shadow DOM, same-origin iframes, SPA mutations, multi-step flows, and framework-controlled inputs, plus a per-site version counter enabling password rotation (12/12 UI-pattern fixtures handled).

Research questions:

RQ1 Does Harbor prevent cross-domain replay and AiTM relay?

RQ2 What is the offline attack cost, and is recoverability free of security cost?

RQ3 What is the per-login runtime overhead versus prior work?

RQ4 How broad is coverage across phishing scenarios and login UIs?

RQ5 Does the Harbor design eliminate the usability defects documented for the prior client-only scheme?

2. Related Work

A. Client-Side Password Transformation

The idea of deriving per-site secrets in the browser dates to PwdHash (Ross et al., 2005), which submitted a hash of the password concatenated with the domain. Two weaknesses limited its protection: the underlying hash function made offline guessing inexpensive, and the transformation ran inside the page, where scripts could read the raw password before it was hashed. AnchorPass (Al-jbour, 2026) revisited the design with a memory-hard KDF, a device-held pepper, and extension-level process isolation (Reis et al., 2019). Harbor keeps the layer at which these systems operate—credentials, with no server involvement—while removing the three properties of AnchorPass that block real deployment (Section 1).

B. Server-Integrated Protocols

OPAQUE (Jarecki et al., 2018) and passkeys/WebAuthn (FIDO Alliance, 2025) provide the strongest available guarantees: the former ensures the server never sees the password and resists pre-computation, the latter replaces shared secrets with public-key credentials. Both, however, require the relying party to change—OPAQUE alters the authentication protocol and the stored verifier; passkeys demand registration ceremonies, credential storage, and cross-device synchronization (Bonneau et al., 2012). Harbor targets the population of sites that have not made, and may never make, those changes.

C. Warnings, Managers, and Mental Models

Field studies show that security warnings suffer habituation and misinterpretation (Akhawe & Felt, 2013; Felt et al., 2016), and that password managers, while reducing reuse, still depend on users heeding domain-mismatch prompts they sometimes override (Gaw & Felten, 2006). Research on mental models finds that users hold incomplete beliefs about how protection mechanisms work (Wash, 2010; Ion et al., 2015), an argument for automatic, transparent protection over designs that require active judgment. Harbor's always-present credential bar follows that principle.

D. Key Recovery

Non-recoverable device secrets are a known adoption hazard. Deterministic mnemonic seed encodings (e.g., BIP-39-style word lists and checksummed codes) are the standard remedy in other domains for turning a high-entropy secret into a recoverable artifact. ROBD applies this idea to origin-bound password derivation, which to our knowledge has not been done before.

3. Threat Model

We scope Harbor to the authentication moment: an adversary who operates a convincing phishing page or a live relay proxy must not obtain a credential usable at the legitimate origin (Bonneau et al., 2012; Verizon, 2024). Our attacker and trust assumptions follow the model introduced for AnchorPass (Al-jbour, 2026); we restate them in our own terms below and note where Harbor extends the model (scheme separation and hosted-subdomain tenancy).

A. Attacker Capabilities

AC1 Control a look-alike site at $D' \neq D$, including its full HTML/JS stack; the deception may use typosquatting, combosquatting, deceptive subdomains, or homograph/IDN confusables (Reynolds et al., 2020; APWG, 2024). **AC2** Proxy a victim's session to the legitimate site in real time (AiTM credential relay) (Microsoft Threat Intelligence, 2023; Kondracki et al., 2021). **AC3** Run large-scale offline guessing against captured tokens on GPU hardware (Biryukov et al., 2021). **AC4** Deliver persuasive lures over

email, SMS, or social channels. **AC5** Present a valid HTTPS certificate for D'. **AC6** Replay credentials leaked elsewhere (credential stuffing) (Thomas et al., 2019).

Out of scope. As with passwords, passkeys, and every client-side scheme we are aware of, we do not defend against a compromised endpoint (keyloggers, screen scraping), a compromised browser process, or theft of an already-established session; these threats require orthogonal mitigations.

B. Trust Assumptions

TA1 The browser correctly enforces site/process isolation and the same-origin policy (Reis et al., 2019). **TA2** The origin the browser reports is the TLS-authenticated endpoint. **TA3** Argon2id (Biryukov et al., 2021), HKDF-SHA256 (Krawczyk & Eronen, 2010), and AES-GCM meet their standard security definitions. **TA4** No malware is present on the device at derivation time.

C. Security Properties

P1 (origin unlinkability). For identical user inputs and distinct scopes $s \neq s'$, tokens T_s, T_s, are computationally independent: they are HKDF expansions of RK under distinct *info* strings, indistinguishable from independent random values under the PRF security of HMAC-SHA256. **P2 (offline hardness).** Recovering P from any set of captured tokens requires evaluating Argon2id per (P-guess, S-guess) pair; with the 128-bit seed S unknown and never transmitted, this is $\geq 2^{H(P)+128}$ memory-hard evaluations. **P3 (forward independence).** Compromise of one site's token or stored hash gives no advantage against any other site (HKDF one-wayness), so P2 holds per-site even under multi-site capture.

4. System Design

A. The Credential Bar

Harbor's chrome renders a credential bar directly below the address bar, in the browser process, outside any web content area (Figure 2). Three consequences follow. (i) *Unspoofable placement:* the bar is rendered in browser chrome, outside the web content area, so no page can draw over or imitate it—closing the popup-spoofing surface an extension inherits. (ii) *No recall burden:* the bar is always present and highlights when the page reports a login form, eliminating AnchorPass's worst usability finding, namely that users must remember to invoke the extension. (iii) *The master password never reaches a page:* credentials are typed into chrome; only a derived token is injected.

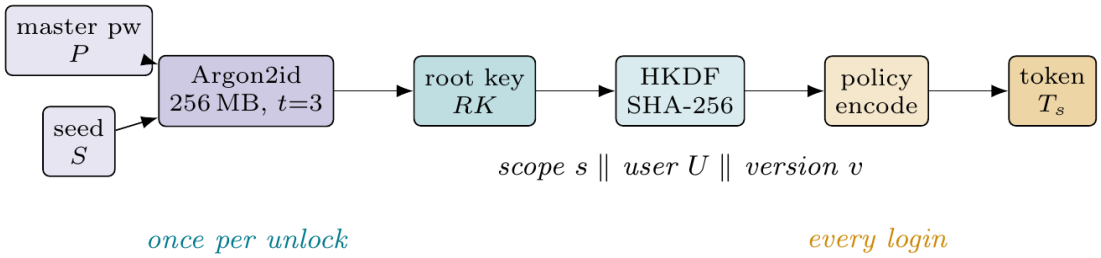

Figure 1*. ROBD derivation pipeline. The memory-hard step (violet inputs → teal RK) runs once per unlock; each login is a fast HKDF expansion keyed by the per-site info string, then a deterministic policy-aware encoding into the amber token. Colour marks the amortization boundary: teal = per-session, amber = per-login.*

B. ROBD: Recoverable Origin-Bound Derivation

Given master password P, recovery seed S (128 bits), scope s, username U, and per-site version v:

$$RK = \text{Argon2id}(P, \text{salt} = H(\text{tag} \parallel S);\ 256\ \text{MB},\ t = 3,\ p = 1) \quad [\text{once/unlock}]$$
$$K_s = \text{HKDF-SHA256}(RK, \text{info} = \text{tag} \parallel s \parallel U \parallel v) \quad [\text{per login}]$$
$$T_s = \text{Encode}(K_s, \text{site policy}) \quad [\text{deterministic}]$$

The recovery code is a base32 encoding of S with a 16-bit checksum (29 characters, grouped for legibility); mistyped codes are rejected before any derivation. Scope s is the registrable domain (eTLD+1) by default, with two refinements below.

C. Scope Computation

Binding to eTLD+1 unifies legitimate subdomains (accounts.google.com and mail.google.com both map to google.com) while keeping attacker registrable domains separate (google.com.evil.io → evil.io). Two refinements preserve security. *Hosted-subdomain platforms:* on platforms where eTLD+1 spans mutually distrusting principals (user1.github.io vs. user2.github.io), the Public Suffix List's private section (Mozilla Foundation, n.d.) is consulted; when it yields a longer registrable domain than the public-only lookup, the full host is used, so distinct tenants derive distinct tokens. *Scheme separation:* HTTP pages derive under a distinct insecure!-prefixed scope, so an HTTPS-to-HTTP downgrade never reveals the HTTPS token. Table 1 summarizes the rule cascade.

***Table 1**. Scope computation as a first-match rule cascade.*

Rule (first match wins)	Example host	Derived scope	Effect
1. Scheme is HTTP	http://bank.com	insecure!bank.com	downgrade never reveals HTTPS token
2. PSL-private / override	user1.github.io	user1.github.io	hosted tenants stay isolated
3. Default: eTLD+1	mail.google.com	google.com	legitimate subdomains unify
3. (attacker domain)	google.com.evil.io	evil.io	deceptive subdomains stay separate

Security trade-off of registrable-domain scoping. Coarsening the binding from the full origin to eTLD+1 is not a pure win, and we state the regression explicitly: it widens the blast radius of a compromise *within* the registrable domain. Under AnchorPass's full-origin binding, a token exfiltrated through a cross-site-scripting flaw on, or a takeover of, one subdomain (e.g., blog.example.com) was useless at accounts.example.com; under eTLD+1 scoping, the same captured token is valid anywhere in example.com. We accept this regression deliberately, for three reasons. First, the web platform already concentrates trust at the registrable domain: cookies may be scoped with Domain=example.com, and mainstream password managers autofill stored credentials across sibling subdomains by default, so an attacker holding one subdomain typically gains equivalent credential access against the deployed alternatives as well—full-origin binding was stricter than the surrounding platform, not matched by it. Second, the two failure modes are not symmetric in likelihood or consequence: the regression requires an active compromise of the legitimate site's own infrastructure (which our attacker model AC1 does not grant, and which independently defeats session cookies regardless of how passwords are derived), whereas the correctness failure it repairs—distinct tokens across a site's own login hosts—occurs on essentially every multi-subdomain deployment and, as AnchorPass's own analysis notes, pushes users to disable protection entirely. An expected-loss comparison therefore favors the coarser scope. Third, Harbor retains a per-site escape hatch: the same full-host override mechanism used for PSL-private platforms lets a user or administrator pin any individually sensitive host back to full-origin binding, and hosted-subdomain platforms where eTLD+1 spans mutually distrusting principals are separated automatically. The residual exposure is sites that segregate mutually distrusting content on sibling subdomains without a Public Suffix List private-section entry; registering such boundaries in the PSL is the ecosystem's established remedy, and we return to this dependency in Section 7.D.

D. Policy-Aware Token Encoding and Rotation

Rather than a fixed truncation of the derived key (as in AnchorPass), ROBD deterministically encodes K_s into a string satisfying a per-site policy (length, required character classes). Required classes are guaranteed at key-derived positions, so a strict composition rule never forces a re-roll and the encoding remains a pure function of (K_s, policy). The per-site version v enters *info*; a Rotate action bumps v during the site's change-password flow, yielding a fresh token while the master secret and seed are untouched—supporting forced rotations and post-compromise recovery.

E. Recovery Seed at Rest, and the Credential Firewall

On this device, S is sealed twice: a PIN is stretched with Argon2id to an AES-256-GCM key wrapping S, and the wrapped blob is additionally sealed with the OS keychain (DPAPI/Keychain/libsecret). Crucially, none of this is load-bearing for recovery: the recovery code reconstructs S on any device, so loss of this file, this PIN, or this machine is survivable.

The credential firewall guards the residual mistake where a user types the master password into a page. On unlock, the trusted process arms a salted fingerprint of P; the page preload hashes password-field contents with the same salt on change/submit and reports the digest. A match blocks the submission, clears the field, and raises a chrome-level alert. Because the fingerprint salt is per-document state in the sandboxed preload, it is re-delivered to every document and frame on load, so the firewall stays armed across navigations. The page never learns P; it only ever computes a salted hash of what the user already typed into it.

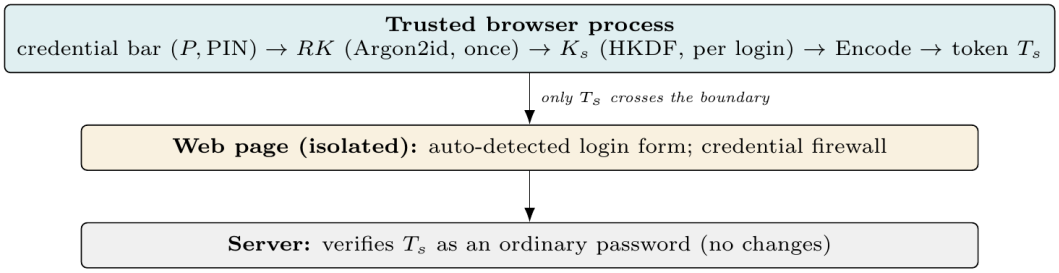

Figure 2*. Harbor dataflow and trust boundary. Raw secrets and RK never leave the browser process; the page and server see only the derived token.*

F. Interface Walkthrough

Figure 3 labels the two rows of Harbor's chrome. The navigation row carries back/forward/reload, an address field with an explicit lock indicator (green "Secure" on HTTPS, red "Not secure" on HTTP), and a vault-status pill ("No vault" → "Locked" → "Unlocked"). The credential bar below it is the security-relevant surface. At its left sits the berth plaque: a badge showing the registrable domain the token will be bound to, lit amber on a normal secure site, outlined red on an insecure one, and flashed

green immediately after a successful fill. Small chips beside it flag a first visit or a non-HTTPS connection. The plaque is the user's anti-phishing anchor: on a look-alike domain it names the look-alike, and the password derived there will not authenticate at the real site.

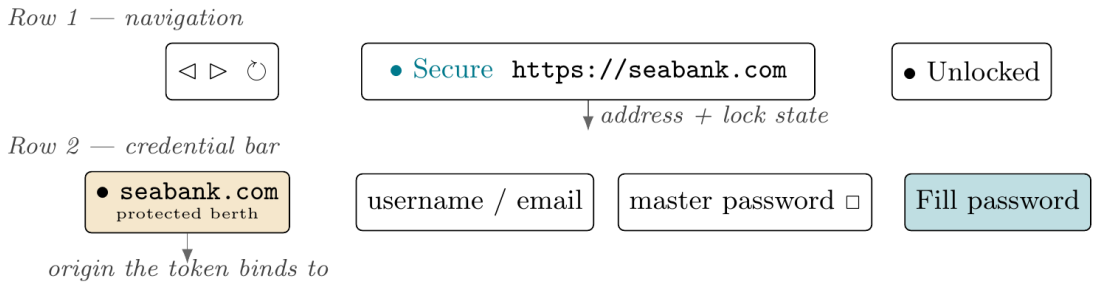

***Figure 3**. Anatomy of Harbor's chrome. The navigation row (top) behaves like any browser; the credential bar (bottom) is where the master password is entered and per-site tokens are derived. The berth plaque names the origin the token is bound to.*

The remaining controls are state-driven (Figure 5). With no vault, the bar offers Set up and Restore. When locked, it shows a PIN box and Unlock. When unlocked and the page reports a login form, a username field, a master-password field (with show/hide), and a Fill password button appear; on an account-creation page the button becomes Create & fill password and both the password and confirm-password fields are filled with the freshly generated per-site password. When unlocked with no form present, the bar shows a calm ready state rather than empty inputs. Crucially, the master password is only ever entered in this chrome surface—never in the page—so the mistake the whole system defends against (typing a secret into the wrong place) is confined to a value that is useless off-origin.

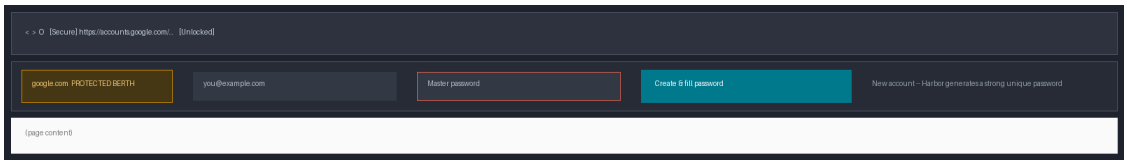

***Figure 4**. The running Harbor browser: the native credential bar sits directly below the address bar on a live HTTPS site, deriving and filling a per-site token while the master password stays in trusted chrome.*

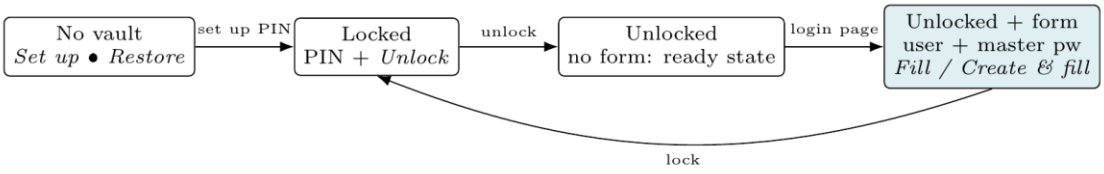

***Figure 5**. Credential-bar states. The bar shows only the controls relevant to the current state; the master-password and Fill controls appear only when the vault is unlocked and the page reports a login or sign-up form.*

G. Universal Login-UI Handling

The page preload runs in an isolated world in every same-origin frame and detects credential fields by traversing shadow DOM, watching SPA mutations, and scoring username candidates by autocomplete, name/id, ARIA, and DOM proximity to the password field. It supports multi-step (username-first) flows via a bounded pending-fill window, avoids hidden honeypot fields, and fills using the native value setter with synthetic input/change events for framework-controlled inputs. Before touching the DOM it re-verifies location.origin against the origin committed at derivation time, closing a navigate-between-derive-and-fill race.

H. Deployment Model

Harbor's output is indistinguishable, from the relying party's perspective, from a user-chosen password: sign-up stores whatever hash the site already uses, and sign-in compares as before. Nothing in the site's protocol, storage schema, or operations changes, so individual users and organizations can adopt Harbor unilaterally and roll it out one account at a time.

5. Implementation

Harbor is a prototype built on Electron/Chromium; we use "browser" as shorthand for this Chromium-based shell, whose chrome we control. The browser process owns navigation, key management, and IPC; the page runs in a separate WebContentsView with context isolation. The master password, seed, and RK exist only in the browser process and are zeroized on lock and on browser close; an idle timer auto-locks. The crypto core uses Argon2id (WebAssembly) and Node's HKDF/AES-GCM. All security-relevant logic (derivation, scope computation, keystore, detector) is factored into modules with unit tests; the experiment harness imports the same modules used at runtime, so reported numbers reflect production code paths.

6. Experimental Evaluation

All measurements below were produced by the bundled harness on an AMD Ryzen 7 8745HS laptop (Radeon 780M) with 28 GB RAM running Windows and Node 24. ROBD parameters: root Argon2id 256 MB/$t = 3/p = 1$; per-site HKDF-SHA256; default token 24 characters over four classes. We report six experiments (E1–E6) answering RQ1–RQ5; E5 (UI compatibility) and E6 (recovery) are new relative to prior work.

A. E1 — Domain Binding (RQ1)

For 1,000 random registrable-domain pairs (s, s') we derive site keys with identical (U, P, S) and measure normalized Hamming distance and collisions (Table 2). Keys are near-independent (mean ≈ 0.50, the expected value for independent random

bitstrings) with zero collisions and 1,000 distinct tokens, supporting P1. Unlike AnchorPass's report of a ~0.98 mean distance, ROBD keys are compared as raw HKDF output, whose expected inter-key distance is 0.5; the security-relevant facts are zero collisions and full independence, not proximity to 1.0. (A value near 1.0 would indicate near-complementary, not independent, outputs.)

***Table 2**. E1: domain binding over 1,000 registrable-domain pairs.*

Metric	Value
Mean normalized Hamming distance (site keys)	0.4997
Standard deviation	0.0311
Minimum / Maximum	0.3945 / 0.5820
Key or token collisions	0
Distinct tokens	1000/1000

B. E2 — Replay and AiTM (RQ1)

For 15 phishing techniques (typo-/combo-squatting, subdomain deception, homograph/IDN, TLD swap, HTTP downgrade, port tricks, lure portals, and an AiTM relay proxy) we derive the token a victim would type on the phishing page, then relay it to a server that stores the hash of the token registered at the legitimate scope. Across 1,005 attempts, 0 succeeded (0.0%) (Table 3); a control login from the legitimate scope verifies correctly. Because binding uses scheme-tagged eTLD+1, every variant maps to a scope distinct from the target.

***Table 3**. E2: cross-domain replay and AiTM relay (representative variants; 1,005 total attempts).*

Variant	Phishing host	Derived scope	Replays
Typosquat	examp1e-bank.com	examp1e-bank.com	0
Combosquat	example-bank-secure.com	example-bank-secure.com	0
Subdomain deception	example-bank.com.attacker.io	attacker.io	0
Homograph (IDN)	xn–exmple-bank-8bb.com	xn–...	0
TLD swap	example-bank.net	example-bank.net	0
HTTP downgrade	example-bank.com	insecure!example-bank.com	0
Port trick	example-bank.com.evil.dev:8443	evil.dev	0
AiTM relay proxy	example-bank.proxysite.app	proxysite.app	0
Total (15 variants)			0/1005

C. E3 — Latency (RQ3)

Table 4 reports warm-trial latencies (visualized in Figure 6). Harbor pays a one-time unlock cost (Argon2id 256 MB) and then derives each login in 0.05 ms mean, versus approximately 118 ms per login for AnchorPass's per-login Argon2id and approximately 12 ms for PwdHash-PBKDF2. The unlock cost is incurred once per browser session (or after an idle auto-lock), not per login, so interactive sign-in is effectively instantaneous. The security-relevant offline cost is unchanged (E4): the memory-hard step still gates every password guess.

Table 4. *E3: latency (warm trials). Harbor amortizes memory-hardness to one unlock per session. The AnchorPass row is reference-machine Argon2id at the prior work's parameters; AnchorPass reports 118 ms on its own hardware. The qualitative gap (per-login memory-hard vs. per-login HKDF) is hardware-independent.*

Operation	Mean (ms)	Median (ms)	SD (ms)
Harbor session unlock (Argon2id 256 MB, once)	621.83	620.45	8.21
Harbor per-login (HKDF + encode)	0.05	0.04	0.04
AnchorPass per-login (Argon2id 128 MB, $t = 2$)	202.66	202.10	2.92
PwdHash-PBKDF2 per-login (100k)	11.77	11.58	0.35

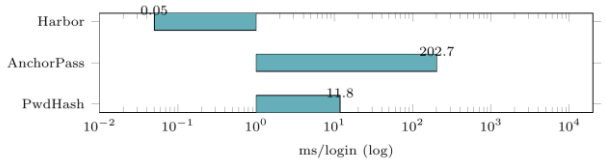

Figure 6*. Per-login latency (measured, log scale). Harbor amortizes the memory-hard step to one unlock, making each login an HKDF expansion approximately 4000× faster than AnchorPass's per-login Argon2id.*

D. E4 — Offline Attack Cost (RQ2)

We consider an adversary equipped with eight NVIDIA RTX 4090 cards who has captured one or more Harbor tokens (or their server-side hashes). Because each Argon2id evaluation at ≥128 MB must stream its working set through GPU global memory, guessing throughput is bounded by memory bandwidth rather than by compute. Following published analyses of high-memory Argon2 (Biryukov et al., 2021; Biryukov et al., 2016; Aumasson et al., 2015), and consistent with the model used for AnchorPass, we credit the adversary a generous 10^4 guesses per second per card (8×10^4 in total).

Table 5 (plotted in Figure 7) shows that without the seed a weak (30-bit) password falls in hours, whereas the 128-bit seed lifts the effective search to $2^{H(P)+128}$, i.e. approximately 10^{35} GPU-years for the same password—computationally

infeasible. Because recoverability is provided by the seed itself (not by weakening it), P2 is preserved: recoverability is free of security cost. P3 (HKDF one-wayness) makes these bounds hold per-site even under multi-site token capture. Since the master password is the one human-chosen input to the bound, setup enforces a strength floor: candidates are checked against a common-password blocklist and simple sequence/repeat patterns, and rejected unless they reach either 12+ characters with mixed classes or a 4-word (≥16-character) passphrase, with a live strength meter guiding the choice. This raises the worst-case $H(P)$ that the offline bound is evaluated against.

***Table 5**. E4: offline cost. 8 ×RTX 4090, approximately 10^4 Argon2id (≥128 MB) guesses/s/GPU.*

Password entropy $H(P)$	**Without seed (2^H)**	**With 128-bit seed (2^{H+128})**
30 bits (weak)	3.7 h	1.5×10^{35} GPU-years
40 bits (medium)	159 days	1.5×10^{38} GPU-years
60 bits (strong)	4.6×10^8 yr	1.6×10^{44} GPU-years

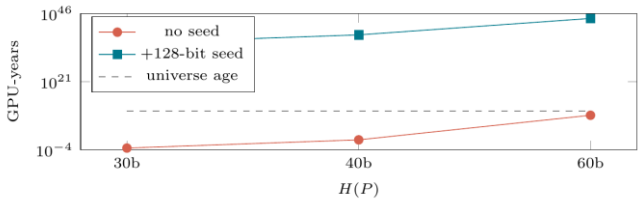

***Figure 7**. Offline cracking cost (analytic). The 128-bit seed (teal) lifts offline cost astronomically above the seedless case (coral) for every password strength; even a weak password with the seed sits far above the age of the universe (dashed).*

E. E5 — Login-UI Compatibility (RQ4)

We evaluate the detector on 12 representative login-UI patterns: classic forms, email/phone logins, autocomplete-annotated fields, formless SPA logins, multi-step username-first and password-only pages, noisy pages with decoy inputs, hidden honeypot passwords, label-free minimal forms, legacy table layouts, and uid-style fields. The detector correctly identified the intended fields on 12/12 patterns, including selecting the real password field over a hidden honeypot. This directly addresses the requirement to manage diverse login UIs, in contrast to AnchorPass's reliance on a single input[type=password] selector. An at-scale measurement of the same detector on live production login pages is planned as future work (Section 8).

F. E6 — Recovery Correctness (RQ2)

We simulate device loss 200 times: generate a seed, render its recovery code, decode it on a fresh "device," and check that every derived site token matches. Seeds round-tripped exactly in 200/200 trials, and a full-pipeline check (Argon2id → HKDF → encode before and after restore) produced identical tokens. In contrast, AnchorPass pepper loss invalidates all tokens with no recovery path.

G. Comparison to Baselines

Table 6 situates Harbor against standard passwords, password managers, PwdHash, passkeys, and AnchorPass. Harbor matches passkey-level phishing/AiTM resistance and AnchorPass's zero-server-change property, while uniquely providing recoverability and sub-millisecond per-login cost, and resolving multi-subdomain correctness.

Table 6*. Security, deployability, and operational properties across approaches.*

Property	Std. pwd	Pwd mgr	Passkeys	AnchorPass	Harbor
Phishing resistance	no	heuristic	crypto	crypto	crypto
AiTM relay resistance	no	no	yes	yes	yes
Offline attack cost	low	low	n/a	high	high
Server changes	none	none	required	none	none
Per-login cost	~0	~0	low	~118 ms	0.05 ms
Secret recoverable	n/a	yes	varies	no	yes
Multi-subdomain correct	n/a	yes	yes	no	yes
Master pw enters page	yes	no	no	no	no(+firewall)

7. Usability and Security Discussion

A. Removing the Prior Work's Usability Defects (RQ5)

AnchorPass's headline usability finding was the recall burden: users must remember to invoke the extension. Moving the credential bar into browser chrome removes the

invocation step entirely and removes the spoofable-popup surface. The two learnability barriers AnchorPass flagged—the unfamiliar term "pepper" and the "one password everywhere" mental-model mismatch—are addressed by user-facing language ("recovery code," "a unique password for this site") and by the always-visible per-site scope in the bar. We report these as design resolutions; a controlled user study measuring behavior remains future work (Section 8).

B. Expert Usability Evaluation

To test whether the design resolutions of Section 7.A hold up under systematic inspection, we conducted an expert usability evaluation of the running Harbor prototype using the two formative methods applied to AnchorPass (Al-jbour, 2026): a cognitive walkthrough (Wharton et al., 1994) of first-time tasks, and a heuristic evaluation against Nielsen's ten usability heuristics (Nielsen, 1994). Unlike a purely manual inspection, the walkthrough was *instrumented*: each task was executed against the live application through a scripted, replayable protocol (included with the artifact) that captures a screenshot and a structured log entry at every step, so every observation below can be reproduced with one command. Five first-time tasks were walked: **CW1** vault setup (master-password strength floor, PIN, recovery-code capture); **CW2** first login on a live production site (github.com/login); **CW3** account creation with Create & fill on a sign-up form; **CW4** restore on a second, fresh profile using only the recovery code; and **CW5** mental-model support from the in-context help. At each step the standard walkthrough questions were assessed, and issues were recorded with the severity scale of the prior work (Critical / Major / Minor / Cosmetic).

What the walkthrough confirmed.

The master-password strength floor operates as specified: "password123" was rejected as too common, a mediocre ten-character password was rated Fair and refused at vault creation with actionable guidance, and a four-word passphrase was accepted as Strong (CW1). The recovery code was displayed once as a checksummed, grouped 29-character string together with an explicit anti-phishing warning ("Harbor will never ask for this code on a web page"). On a live production login page, the berth plaque named the registrable domain (github.com, "protected berth"), the first-visit chip appeared, and Fill password injected a 24-character four-class token into the site's real password field with visible success feedback (CW2). The sign-up detector switched the action to Create & fill and filled both password fields with identical fresh tokens (CW3). On the fresh profile, a deliberately mistyped recovery code was rejected before any derivation with an explicit checksum message, and after a correct restore the token derived for the same site and username was bit-identical to the first device's — a live, end-to-end confirmation of E6 on a production site (CW4). The nine-item in-context help explains the master password, per-site tokens, the plaque, and the recovery code in plain language, with no exposure of internal terms such as "pepper" (CW5).

Issues found and their resolution.

Table 7 lists every issue identified. The single Major finding is the kind that motivates instrumented evaluation: the credential firewall's fingerprint salt was delivered only to the document loaded at the moment of unlock, so on every page visited afterwards the firewall was silently inactive — typing the master password into a page after any navigation produced no alert and left the secret in the page's field. The defect is invisible in ordinary demonstration (the firewall works on the unlock-time page) and was surfaced only by walking the realistic sequence unlock → browse → mistype. It was fixed by re-delivering the salt to every frame on document load, and the fix was re-verified live: post-navigation, the alert fires and the field is cleared. Two Minor issues (raw IPC prefixes leaking into user-facing error messages; Harbor's own bundled welcome page displaying "Not secure," contradicting trust cues on the very first screen) were likewise fixed and re-verified in the evaluated build. The remaining observations are retained as input to the planned user study.

Table 7. Expert-evaluation findings on the running prototype, with resolutions.

ID	Finding (observed live)	Resolution / status	Severity
HE-1	Credential firewall armed only on the unlock-time document; after any navigation, typing the master password into a page produced no alert and no field clearing	Fixed: fingerprint salt re-delivered to every document and frame on load; block and field-clear re-verified after navigation	Major
HE-2	Internal IPC prefixes leaked into user-facing errors (e.g., "unlock': Error: incorrect PIN")	Fixed: messages sanitized ("incorrect PIN"); re-verified	Minor
CW3-1	Harbor's own bundled welcome page displayed "Not secure" / "unprotected," contradicting trust cues on the first screen a user sees	Fixed: bundled pages now display as secure content; re-verified	Minor
CW1-1	"Copy" places the recovery code in the system clipboard, in tension with the offline-custody guidance shown beside it	Noted; a clipboard-clear timer is a planned refinement	Minor
CW4-1	After restore on a new device, the "first visit" chip reappears on familiar sites (the visited list is per-device)	Retained by design (it is this device's first visit); flagged for the user study	Cosmetic
HE-3	Success messages auto-dismiss after ~3 s; fill confirmation can be missed, though the plaque's green flash remains	Cosmetic; a persistent per-site "filled" indicator is a planned refinement	Cosmetic

Comparison with the prior work's defects.

None of the four usability defects documented for AnchorPass recurred. The Major recall burden (users must remember to invoke the extension) is removed: on both

the live site and the practice page, the bar surfaced the credential controls automatically when a login or sign-up form appeared, with no invocation step (CW2, CW3). The spoofable-popup surface is removed by construction, since the credential surface is browser chrome that page content cannot draw over. The "pepper" terminology barrier does not recur: the interface and help speak only of the master password, per-site passwords, and the recovery code (CW1, CW5). The "one password everywhere" mental-model mismatch is countered by the always-visible per-site scope in the berth plaque and by help copy stating that each website gets its own different password (CW5). The evaluation also *confirmed* the operational trade-off analyzed in Section 7.E: unlocking with an incorrect master password succeeds silently (the vault shows Unlocked; no warning is possible without a stored verifier), and the resulting mismatch surfaces only at the site — underscoring why that subsection treats error displacement explicitly.

Scope of this evidence.

Expert inspection — even instrumented and replayable — identifies interface and implementation problems but does not measure user behavior, comprehension under attack, or habituation; those require the controlled user study that remains the most important next step (Section 8). Within that scope, the evaluation supports RQ5: the specific defects documented for the prior client-only scheme do not recur, one Major implementation defect in a headline security feature was caught and fixed before any deployment, and the residual operational burdens are identified and discussed in Sections 7.D and 7.E.

C. Why Recoverability Does Not Weaken Security

The seed is an additional 128-bit secret that (i) never leaves the device in normal operation, (ii) is not transmitted to any site, and (iii) is only ever surfaced to the user as a code they store offline. Its presence in the derivation gives the +128 bits; its recoverability is a property of encoding, not of exposure. An attacker gains nothing unless they obtain the code, which is never entered into a web page and is protected at rest.

D. Residual Risks

Endpoint and browser-process compromise remain out of scope, as for all client schemes and passkeys. A user who ignores the bar and manually types a site-specific password gets no protection for that account (the master password itself is caught by the firewall). The recovery code is a phishable secret in principle; extending the firewall's fingerprint set to cover the code is straightforward, and onboarding should stress that Harbor never requests the code on a page. eTLD+1 scoping trusts the Public Suffix List (Mozilla Foundation, n.d.); the private-section refinement handles hosted-subdomain platforms, but PSL currency must be maintained, and the intra-registrable-domain blast-radius regression analyzed in Section 4.C remains for sites that host mutually distrusting principals on sibling subdomains without a PSL entry (the per-site full-host override is the local mitigation).

E. Operational Scenarios

Statelessness—no per-site secrets are stored anywhere—is the source of Harbor's recoverability and breach-resistance, but it also shapes several user-facing behaviors that deserve explicit treatment.

Wrong or inconsistent master password fails silently, at the site. Because no verifier of the master password is stored, Harbor cannot detect a mistyped or different master password: any input deterministically yields some well-formed token, and the failure surfaces only as the site rejecting the login. Nothing is lost—re-entering the correct master password regenerates the correct token—but the error is displaced from the trusted UI to the relying party, which may confuse users. The obvious mitigation, a persistent salted fingerprint of the master password used to warn "this does not match your usual master password," carries a real cost: a fingerprint at rest gives an attacker with device access an offline oracle for master-password guesses that is independent of the seed, collapsing the offline bound from $2^{H(P)+128}$ to $2^{H(P)}$ for that attacker. Harbor therefore keeps the credential-firewall fingerprint session-only (armed at unlock, cleared at lock); a persistent typo-check is offered, if at all, as an explicit opt-in with this trade-off stated.

Multiple master passwords are possible but pointless. Determinism imposes only per-site consistency: whichever master password was used when a site's token was first registered must be used for every subsequent login there. A user may thus operate distinct master passwords for distinct sites, and all flows function. However, this recreates exactly the memory burden the scheme removes—n secrets plus an unrecorded site-to-secret mapping, with silent lockout on a forgotten pairing—while adding no per-site unlinkability, which the scope input already provides. The rational configuration is one strong master password; entropy is better added by lengthening it than by multiplying it.

Changing the master password is a global rotation. Because the master password is a derivation input, changing it changes every site's token; the user must then complete each site's change-password flow. The per-site version counter covers the common case (rotating one compromised site) without this cost; a master-password change is the deliberate, heavyweight "rotate everything" action, appropriate after suspected master-password disclosure.

Recovery-code custody. The code and the master password are the two factors of the offline bound; their compromise must be decorrelated. Store the code offline (paper or metal), in a location and medium disjoint from anything holding the master password; duplicate it across two locations to cover loss as well as theft; and never enter it into web content—the trusted bar is its only legitimate input surface. Loss of the code alone is survivable while any enrolled device functions (the code can be re-displayed after PIN confirmation); loss of the code and all devices is unrecoverable by design, since no escrow exists.

Migrating existing accounts. Unlike a storing manager, Harbor cannot carry legacy passwords: each existing account is switched over once, by running the site's change-password flow and filling the new token. This per-site, one-time cost is the price of storing nothing; it also naturally audits the account inventory. Domain migrations by the site itself change the scope and thus the token; the per-site full-host override and version counter give users a manual path, and PSL-informed alias lists are future work.

F. Design-Space Position

Figure 8 places Harbor in the phishing resistance vs. deployability trade-off. We place Harbor above AnchorPass on the resistance axis for three concrete reasons: the credential firewall blocks master-password disclosure to pages, which AnchorPass cannot; the chrome-level bar removes the spoofable extension-popup surface; and scheme-separated scopes close HTTPS-to-HTTP downgrade capture. Its deployability matches AnchorPass and passwords (no server change).

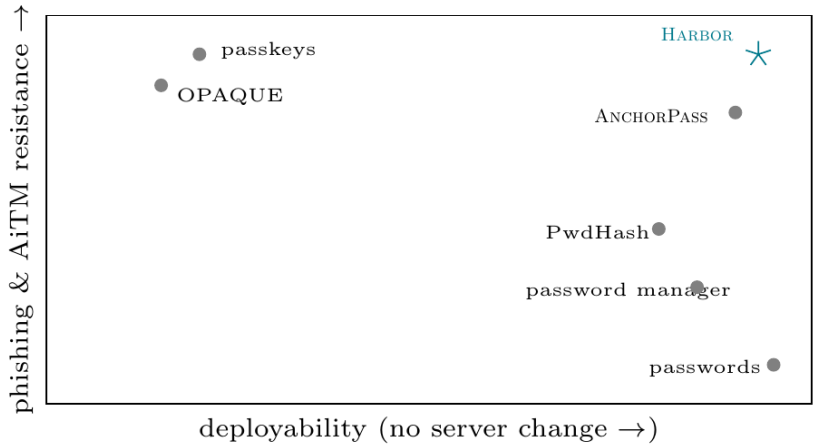

Figure 8*. Design space. Harbor (teal star) sits in the previously empty top-right corner: passkey-level phishing/AiTM resistance with password-level deployability (no server change), improving on AnchorPass in resistance while retaining its zero-server-change property.*

8. Limitations and Future Work

Our security and performance claims are backed by measured experiments and unit tests, and usability evidence now comes from an instrumented expert evaluation (Section 7.B); expert inspection surfaces interface and implementation problems but does not measure behavior, so a diverse-population study measuring task success, comprehension under realistic phishing, and long-term habituation is the most important next step (Reeder et al., 2018; Ion et al., 2015). Additional directions: (i) formal, game-based proofs of P1–P3 and of truncated-output unlinkability; (ii) an at-scale crawl measuring detector success on live login pages (e.g., over a manipulation-

hardened ranking such as Tranco (Le Pochat et al., 2019)) to complement the 12-pattern fixture suite; (iii) standardizing ROBD as a browser-native API with system-level attestation; (iv) multi-device seed portability via OS-keychain escrow or threshold recovery, with a formal loss-vs.-compromise analysis; (v) enterprise management (managed distribution, allow-lists, CT-log pinning); and (vi) mobile ports with biometric unlock.

9. Conclusion

Harbor shows that origin-bound passwords can be made recoverable, fast, and correct on real sites without giving up their defining advantages—credential-layer phishing/AiTM resistance and zero server changes. By replacing the non-recoverable pepper with a recovery seed, amortizing memory-hardness to one unlock, scoping to the registrable domain with public-suffix–aware refinements, and moving the credential surface into unspoofable browser chrome with a credential firewall, Harbor removes the practical barriers that kept prior client-only schemes from deployment. Measured results—0.0% replay success, 0.05 ms per-login derivation, 200/200 exact recoveries, and 12/12 UI patterns handled—indicate a pragmatic path to reduce credential compromise today.

10. Funding

This research received no external funding.

11. Conflict of Interest

The authors declare no conflict of interest.

12. References

Akhawe, D., & Felt, A. P. (2013). Alice in Warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the 22nd USENIX Security Symposium* (pp. 257–272). USENIX Association.

Al-jbour, M. S. (2026). Mitigating phishing attacks with origin-bound password tokens: A client-only approach with zero server changes. In A. Moallem (Ed.), *HCI for Cybersecurity, Privacy and Trust. HCII 2026, LNCS vol. 16738*. Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-032-29723-5_16

Alsharnouby, M., Alaca, F., & Chiasson, S. (2015). Why phishing still works: User strategies for combating phishing attacks. *International Journal of Human-Computer Studies, 82*, 69–82. https://doi.org/10.1016/j.ijhcs.2015.05.005

APWG. (2024). *Phishing activity trends report, 4th quarter 2023*. Anti-Phishing Working Group. https://apwg.org/trendsreports/

Aumasson, J.-P., et al. (2015). *Password Hashing Competition—final report*. https://www.password-hashing.net/

Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: New generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy* (pp. 292–302). IEEE. https://doi.org/10.1109/EuroSP.2016.31

Biryukov, A., Dinu, D., Khovratovich, D., & Josefsson, S. (2021). *Argon2 memory-hard function for password hashing and proof-of-work applications* (RFC 9106). IETF. https://doi.org/10.17487/RFC9106

Bonneau, J., Herley, C., van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy* (pp. 553–567). IEEE. https://doi.org/10.1109/SP.2012.44

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 581–590). ACM. https://doi.org/10.1145/1124772.1124861

Felt, A. P., Reeder, R. W., Ainslie, A., et al. (2016). Rethinking connection security indicators. In *Proceedings of the Twelfth Symposium on Usable Privacy and Security* (pp. 1–14). USENIX Association.

FIDO Alliance. (2025). *Passkeys: Passwordless authentication*. https://fidoalliance.org/passkeys/

Gaw, S., & Felten, E. W. (2006). Password management strategies for online accounts. In *Proceedings of the Second Symposium on Usable Privacy and Security* (pp. 44–55). ACM. https://doi.org/10.1145/1143120.1143127

Ion, I., Reeder, R., & Consolvo, S. (2015). "...No one can hack my mind": Comparing expert and non-expert security practices. In *Proceedings of the Eleventh Symposium on Usable Privacy and Security* (pp. 327–346). USENIX Association.

Jarecki, S., Krawczyk, H., & Xu, J. (2018). OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In *Advances in Cryptology—EUROCRYPT 2018, LNCS vol. 10822* (pp. 456–486). Springer. https://doi.org/10.1007/978-3-319-78372-7_15

Kondracki, B., Azad, B. A., Starov, O., & Nikiforakis, N. (2021). Catching transparent phish: Analyzing and detecting MITM phishing toolkits. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 36–50). ACM. https://doi.org/10.1145/3460120.3484765

Krawczyk, H., & Eronen, P. (2010). *HMAC-based extract-and-expand key derivation function (HKDF)* (RFC 5869). IETF. https://doi.org/10.17487/RFC5869

Le Pochat, V., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., & Joosen, W. (2019). Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society. https://doi.org/10.14722/ndss.2019.23386

Microsoft Threat Intelligence. (2023). *Detecting and mitigating adversary-in-the-middle phishing attacks*. Microsoft Security Blog.

Mozilla Foundation. (n.d.). *Public Suffix List*. https://publicsuffix.org/

Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 25–62). John Wiley & Sons.

Reeder, R. W., Felt, A. P., Consolvo, S., et al. (2018). An experience sampling study of user reactions to browser warnings in the field. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). ACM. https://doi.org/10.1145/3173574.3174086

Reis, C., Moshchuk, A., & Oskov, N. (2019). Site isolation: Process separation for web sites within the browser. In *Proceedings of the 28th USENIX Security Symposium* (pp. 1661–1678). USENIX Association.

Reynolds, J., Kumar, D., Ma, Z., et al. (2020). Measuring identity confusion with uniform resource locators. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). ACM. https://doi.org/10.1145/3313831.3376213

Ross, B., Jackson, C., Miyake, N., Boneh, D., & Mitchell, J. C. (2005). Stronger password authentication using browser extensions. In *Proceedings of the 14th USENIX Security Symposium*. USENIX Association.

Thomas, K., Pullman, J., Yeo, K., et al. (2019). Protecting accounts from credential stuffing with password breach alerting. In *Proceedings of the 28th USENIX Security Symposium* (pp. 1556–1571). USENIX Association.

Verizon. (2024). *2024 data breach investigations report*. Verizon Business. https://www.verizon.com/business/resources/reports/dbir/

Wash, R. (2010). Folk models of home computer security. In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (pp. 1–16). ACM. https://doi.org/10.1145/1837110.1837125

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). The cognitive walkthrough method: A practitioner's guide. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 105–140). John Wiley & Sons.