Reflections on Disregarding Trust

Browser in the Middle (BiTM) — weaponizing browser trust boundaries without Chrome 0-days. EvilRBI reference implementation.

TALK	ATTACK CLASS	IMPLEMENTATION	LICENSE
Reflections on Disregarding Trust	Browser in the Middle (BiTM)	EvilRBI (RBI Proxy), repository ~/proxy	MIT

DEFAULT LISTEN	PUBLIC INGRESS	CHROME EXPLOITS / 0-DAYS
:8879	TunnelTug + public_url (e.g. https://phisheries.dev)	None

Research and authorized assessment use only. Misuse without explicit permission is unlawful and unethical.

A white paper for DEF CON 34
Browser in the Middle (BiTM) · EvilRBI reference implementation · Research and authorized assessment only

Abstract

Modern enterprise security trusts the browser: isolation products stream untrusted pages into a remote engine; component extensions attest devices; WebUI and service workers keep the product running. **Reflections on Disregarding Trust** asks what happens when those same mechanisms are operated *against* the user they were meant to protect.

This paper describes **Browser in the Middle (BiTM)** as distinct from classic adversary-in-the-middle (AiTM) reverse proxies. In BiTM, authentication does not complete in the victim's tab. It completes in an attacker-controlled **middle browser**. The victim sees a faithful DOM/MHTML mirror and supplies clicks, keys, and MFA ceremonies; the middle browser holds the real IdP session and the post-auth cookie jar.

EvilRBI is the reference implementation: a Go proxy that drives Chrome via CDP (chromedp), mirrors upstream documents over WebSocket, relays input and WebAuthn/FastPass, and—on the victim side—abuses **stock Chrome trust surfaces** without installing a malicious CRX and **without Chrome 0-days**. Privilege comes from *who* is trusted (component extensions, error-page contexts, WebUI Mojo, first-party update rails), not from memory corruption.

The system has matured from a single EV magic-RPC foothold into a **multi-rail control plane**. Google Endpoint Verification remains a strong primary path while it lasts. Continuity planning assumes magic may eventually change or close; **inject, apex service workers, first-party components, sticky CDP agents, and chrome:// WebUI fleet control** remain available without treating magic as the only root.

Audience: defenders, authorized red teams, and browser/platform engineers.

Table of contents

1. Thesis and threat model

1.1 One sentence

> **Disregard trust by operating Chrome's own backend—components, deputy contexts, service workers, WebUI Mojo, and CDP residency—not by breaking the renderer.**

1.2 AiTM versus BiTM

	AiTM	BiTM
Sits on	HTTP stream (reverse proxy)	Middle browser (real Chrome)
Captures	Tokens and cookies on the wire	Clicks, keys, credentials, MFA, full browser session
Where auth runs	Still largely in the victim's browser	In the **middle** browser's IdP tab
Victim sees	Proxied IdP HTML at attacker domain	Mirror of the real IdP (Shadow DOM / MHTML)
Typical win	Session cookie / OAuth code	Live upstream session after victim-completed MFA

AiTM tools such as Evilginx excel at stream capture and injection. BiTM tools excel when MFA is device-bound or the goal is a **browser session that already completed WebAuthn or FastPass in a machine the attacker controls**.

EvilRBI is BiTM. The victim is a **display and input terminal**. The middle browser is **authoritative**.

1.3 What is not used

Technique	Used?
Chrome memory corruption / RCE	No
Sandbox escape	No
Unpatched Chrome CVE / 0-day	No
Victim-side malicious CRX install	No
Intended APIs and first-party surfaces	**Yes**
Confused-deputy / weak-origin logic	**Yes**

This is not a Chrome bug report. It is a study of **trust boundaries**: RBI-shaped mirroring, built-in Endpoint Verification, component `externally_connectable` APIs, error-page V8, apex service workers, and `chrome://` internals intended for debugging and fleet management.

1.4 Inversion of Remote Browser Isolation

Defensive RBI isolates untrusted content in a remote engine so malware never reaches the endpoint. EvilRBI inverts the trust direction: the remote engine is the **session owner**; the endpoint is a **mirror**. The same vocabulary—viewport stream, input relay, remote paint—supports the opposite security outcome.

1.5 Scope of “backend”

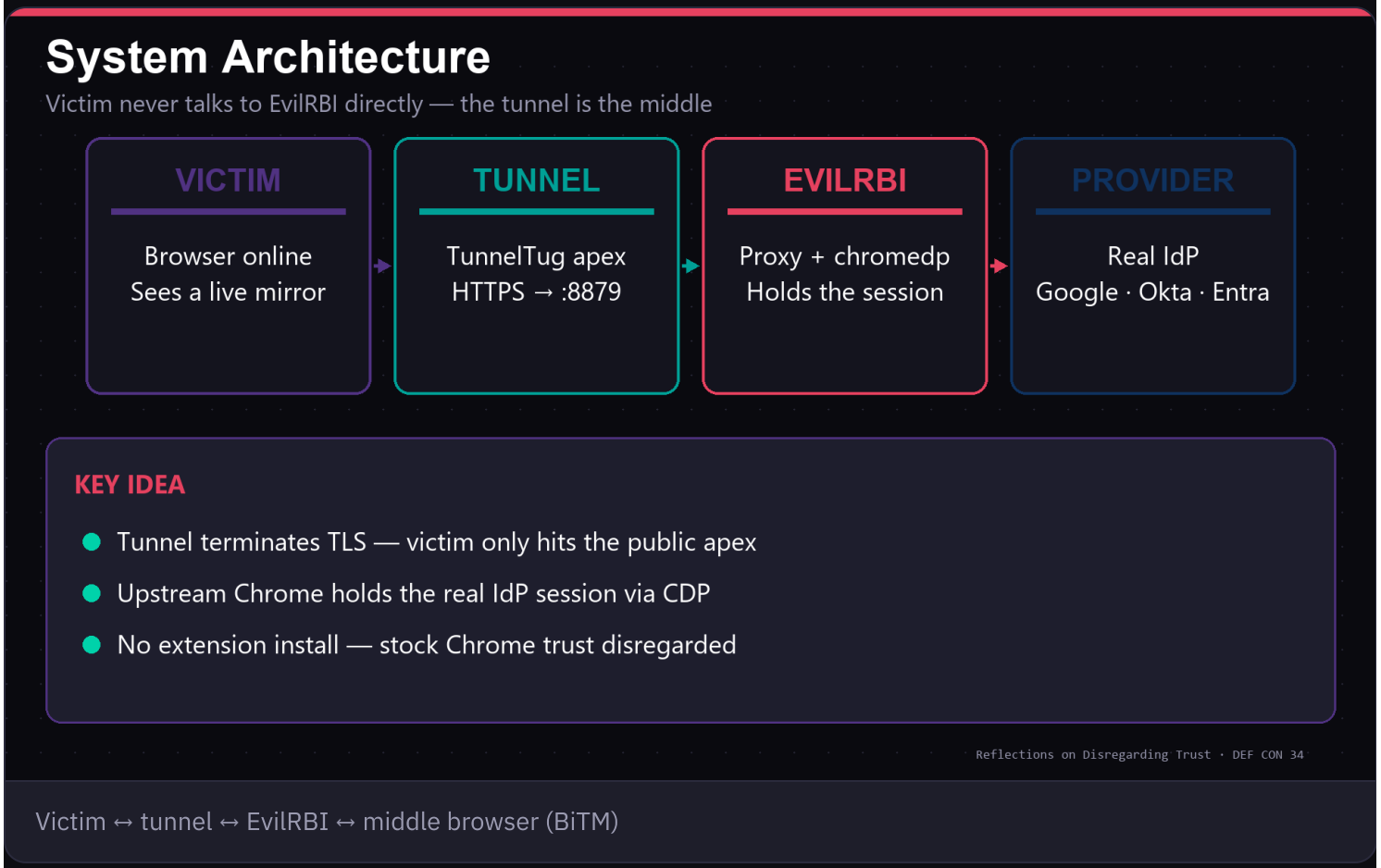
When this paper says the **whole backend of the browser** was used, it means the product surfaces that keep Chrome an enterprise platform:

- CDP and a durable profile (session, cookies, automation)
- Built-in / first-party **component extensions** (EV, Gemini-in-Chrome, Hangouts, Docs Offline, Payments, ...)
- **Confused-deputy** contexts (`chrome-error://chromewebdata/` via `chrome://dino`)
- **Service workers** on the lure origin and fleet control via WebUI

- `chrome://` **Mojo** (web-app-internals for Isolated Web Apps; serviceworker-internals for SW start/stop)
- Google's **Omaha / update2/crx** rail as the supply path for those components

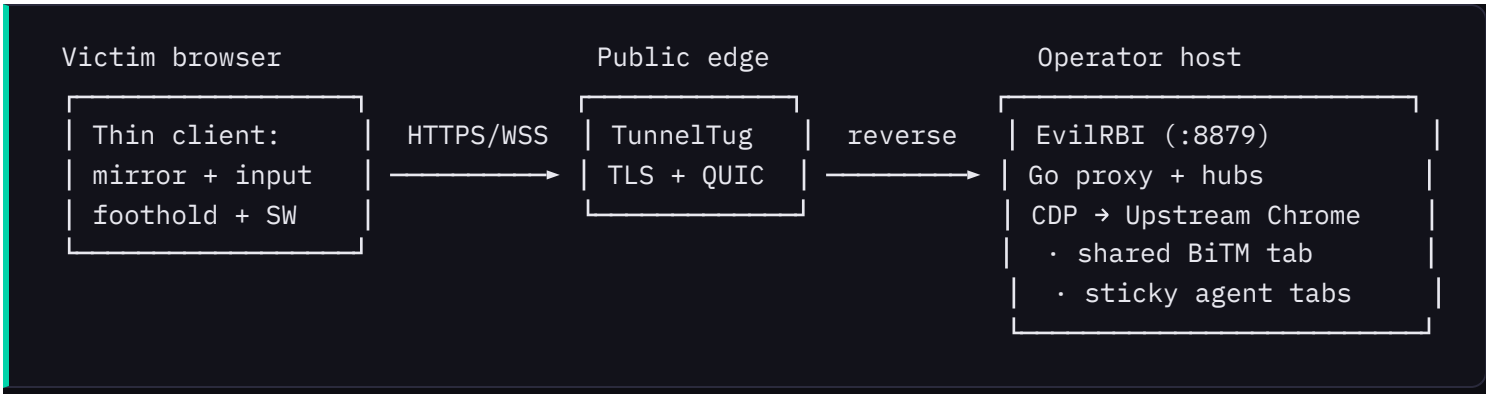
None of that requires a 0-day. All of it requires *operating* mechanisms Chrome already trusts.

2. Architecture



Victim ↔ tunnel ↔ EvilRBI ↔ middle browser (BiTM)

EvilRBI is three cooperating planes plus public ingress.



Role	Process	Owns
Viewer	Victim's stock Chrome	UI, keystrokes, OS passkeys, built-in EV if present
Proxy	`proxy` / `proxy.exe`	Routing, mirror stream, MFA hubs, sticky agents, inventory
Upstream	chromedp Chrome	Real IdP navigations, authoritative cookies, hooked credentials

Two Chrome contexts must stay distinct:

1. **Upstream (operator)** — no attacker CRX required; full CDP. Holds the real login.
2. **Downstream (victim)** — no CRX install; elevated *relative to a normal website* only because of the dino/chromewebdata deputy path and first-party components already present.

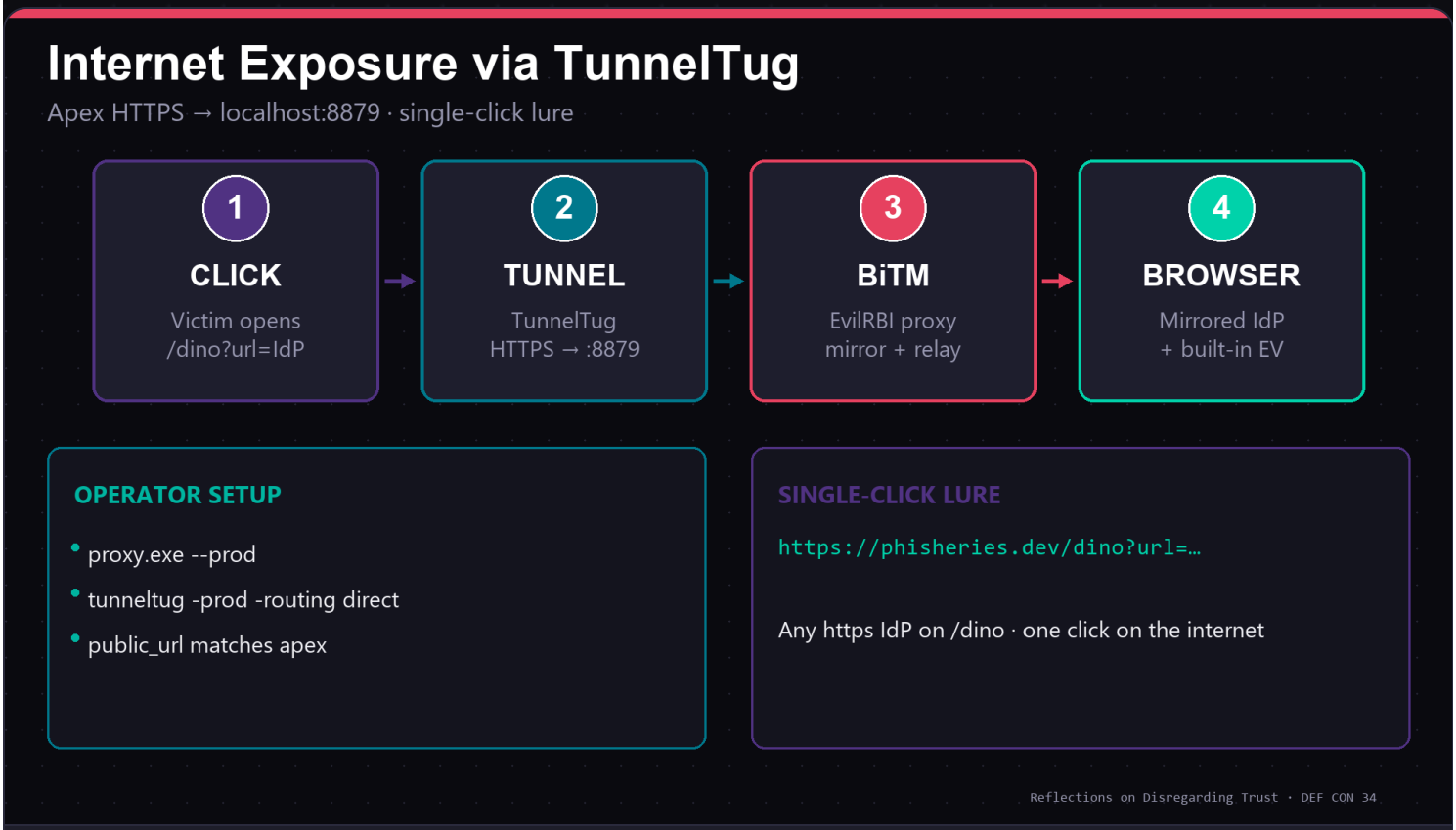
Portal scripts under `src/assets/dino/` are ordinary HTTPS `<script>` tags. Their power is the **context** they run in (deputy + SW + component messaging), not a sideloaded manifest.

2.1 Shared tab versus sticky agents

The **shared upstream tab** is the BiTM mirror: navigate IdP, capture MHTML, dispatch keys. Control-plane work must not thrash that tab mid-login.

Sticky agents are separate background targets (`Target.CreateTarget` , background where possible), re-armed on a timer (~4s after start, then ~75s). Core set: chromewebdata inject, Gemini document for Glic, and `serviceworker-internals` . On demand: Hangouts/Docs origins, `web-app-internals` . Glic and component RPCs prefer sticky tabs so the victim keeps seeing the IdP.

3. Internet exposure



Internet ingress via TunnelTug — one-click /dino?url= lure

EvilRBI listens on localhost by default. Internet victims reach it through **TunnelTug**: public HTTPS (and WebSocket) on a real domain, reverse-proxied over a QUIC/Yamux tunnel to `:8879`.

Typical operator posture:

- `-prod` — ACME TLS on the edge
- `-routing direct` — apex domain maps to one tunnel client (short lure URLs)
- `-local 8879` — forward to EvilRBI

`public_url` in config (e.g. `https://phisheries.dev`) propagates into client config so scripts and WebSockets use the tunnel origin for remote viewers.

3.1 One-click lure

```
https://phisheries.dev/dino?url=https://accounts.google.com/addsession
```

Portal routes (`/dino` , `/blank` , `/persist`) accept `?url=` even under `--prod` route maps. That is intentional for demos and a **policy smell** for production allowlists: the portal is an open stream entry if exposed.

On that click: tunnel TLS → portal shell → chromewebdata foothold arming → upstream navigates the real IdP → stream opens → optional EV pairing scripts load from the lure origin.

---

4. The BiTM data plane

This plane is enough for a complete auth demo. It does not require EV magic.

4.1 Mirror

Upstream `page.CaptureSnapshot` (MHTML) or alternate modes (screencast JPEG, hybrid HTML) produce frames. The server pushes `viewport_frame` on `WS /api/stream-sync` . The client hydrates into a Shadow DOM stage (`rbi-client.js` , `mhtml-hydrate.js`). Hydration strips CSP/sandbox attributes that would block the viewer's shims—necessary for the product, and itself a trust inversion relative to "show only pixels."

4.2 Input

Victim key, fill, focus, and click messages become CDP input events on the **shared** upstream tab. Auth hosts (Google, Okta, Microsoft, AWS, Auth0) use specialized fill and settle paths so SPAs keep up.

4.3 Session phases

Server-owned phases (`init` → `bootstrapping` → `streaming` → `settling` → `authenticating` → `error`) gate input and MFA handoffs. The client remains thin; the proxy owns truth about state.

4.4 Modes

Default mirror mode is **`rbi`** (DOM/MHTML). Other modes (`hybrid` , `mhtml` , `screencast` , `snapshot`) trade fidelity for speed or debug convenience. Headed upstream Chrome is

recommended for OAuth, CAPTCHA, and passkeys.

---

5. Victim foothold: confused deputy without install

A page at `https://evil.example` cannot casually message Google's Endpoint Verification or register an apex-scoped service worker that bridges into component extensions. The trusted-loop portal can—not because the user installed malware, but because Chrome elevates **error and offline surfaces**.

5.1 Dino → chromewebdata

`chrome://dino` is a reliable navigable internal URL. Chrome surfaces the offline game as `chrome-error://chromewebdata/` (often with a network error). That interstitial is a **privileged V8 context** relative to a normal secure origin: the portal parks there, presents extension-origin metadata, and injects agent scripts.

Server-side, the same pattern is used as a **CDP anchor** for cookie and target enumeration (see §10).

5.2 Origin presentation

Without navigating to `chrome-extension://…` (often blocked for web content), the portal **presents** the built-in EV extension origin while the parent remains on chromewebdata. Streamed IdP HTML lives in a child frame; hooking scripts stay in the deputy parent. That split is the confused-deputy core: elevated parent, believable child UI.

5.3 Apex service worker

`/rbi-trusted-loop-sw.js` is served with `Service-Worker-Allowed: /`, so registration can cover the **entire lure origin**. The SW imports bridge code, caches portal assets, and forwards control messages (`GLIC_RPC`, `COMPONENT_RPC`, `OFFSCREEN_MAINTAIN`, `SW_INTERNALS_MAINTAIN`, EV-related RPC) into the proxy over HTTP. On a normal site this would be extraordinary; in the trusted-loop design it is the **always-on hop** between portal JavaScript and operator infrastructure.

5.4 Inventory

Probes catalog dozens of surfaces (page APIs, SW, EV bridge, chromewebdata anchors). Scores change by context: a hot dino foothold scores high; a cold re-probe without EV pairing scores mostly page_v8 + SW. Operators should treat inventory as **capability telemetry**, not a marketing number—and not as proof that magic is the only path.

---

6. Endpoint Verification and the offscreen agent

6.1 Why EV matters

Endpoint Verification (`callobklhcbilhphinckomhgkigmfocg`) ships as an **INTERNAL** component extension: cookies, idle, nativeMessaging, offscreen, identity, enterprise reporting hooks, and host access to `*.google.com` . Enterprise posture often *trusts* signals from this extension. EvilRBI does not install it; it **pairs** with what Chrome already loaded.

6.2 Magic RPC (primary today)

The stock MessageServiceHost path accepts a fixed magic constant and wildcard host:

```
{ "magic": 91556947316803, "host": "*", "method": "hello", "args": [] }
```

From the deputy context, `chrome.runtime.sendMessage` into EV's service worker can invoke methods such as `hello` , `getAccounts` , `getQuirks` , `recordLog` , and related pipeline hooks. That is logic abuse of a first-party control plane—not a renderer exploit.

6.3 Offscreen agent

`offscreen_script.js` (vendored from EV's family of offscreen/agent payloads and extended for portal bootstrap) is injected into chromewebdata via CDP when the confused-deputy window is live. It provides an invoke path, retries through SW and parent relay, and can fall back to proxy `/api/rbi/ev-rpc` stubs when native magic is unavailable.

6.4 Continuity assumption (preplanning, not panic)

EV works for a while; multi-rail is insurance.

Surface	Continuity planning
chromewebdata **inject** of the agent	Likely still works after EV code churn
Magic constant + host `*` acceptance	May disappear on a future EV update
Sticky upstream agents, Glic, Hangouts, WebUI	Independent of victim-side magic

Day-to-day operations should still **use EV hard** while magic pairs. Periodic `RBIBackupFoothold.scan()` is enough to prove parachutes remain packed. Magic death is a future state, not today's default narrative for demos.

7. Operating the browser backend

Beyond EV, research and implementation mapped **first-party components** and WebUI:

Component	ID (stable)	Role in EvilRBI
Endpoint Verification	`callobklhcbilhphinckomhgkigmfocg`	Magic RPC / accounts / quirks (primary)
Gemini in Chrome (Glic)	`admccjkmockfdflocgggjfgdacdodkdf`	`glicPrivate.*` via externally_connectable document
Hangouts services	`nkeimhogjdpnpccoofpliimaahmaaome`	CPU / enterprise hardware / WebRTC logging APIs on Meet
Docs Offline	`ghbmnnjooekpmoecnnnilnnbdlolhkhi`	External rail on docs/drive
CWS Payments	`nmmhkkegccagdldgiimedpiccmgmieda`	Soft presence probe (MV2 app)

Discovery used Omaha **update2/crx** observation, `resources.pak` extraction, and **Dagger/Loki/Helheim** decompile of real extension trees—not CVEs. Helheim is tuned for **component/Mojo surfaces**, not only classic XSS triage.

Glic is especially instructive: its service worker requires a real `documentId` and connectable origin. It has no MV3 Offscreen Document API of its own. The operational "offscreen Glic" is therefore a **sticky https://gemini.google.com/** tab evaluated under CDP—document-backed by design.

8. Magic-optional multi-rail control

`backup_foothold.js` (loaded with the portal alongside `glic_foothold.js`) treats control as a **priority list**, not a single RPC.

Priority	Channel	Depends on magic?
10	Apex trusted-loop SW	No
20	Offscreen / chromewebdata context + proxy hop	No
25	L6 serviceworker-internals fleet	No
30	Glic sticky document	No
40	Hangouts (Meet)	No
50	Docs Offline	No
60	Payments (soft)	No
90	EV magic	**Yes** (optional)

```
await RBIBackupFoothold.scan()
// primary, live_no_magic[], magic_dead, per-channel reports
```

Success without magic is simply: **at least one non-magic rail live**, and a chosen `primary`. When magic still works, EV rows remain high-value; they are not abandoned for ideology.

Portal → MessageChannel → apex SW → proxy HTTP is the same bootstrap shape for Glic, generic components, offscreen maintain, and SW fleet maintain. That sameness is deliberate: one mental model for operators.

9. Sticky agents and WebUI residency (L4–L6, IWA)

9.1 Durability layers

Layer	Idea	Status
L0	EV magic pairing	Primary while available
L1	chromewebdata + agent inject	Sticky + reinject
L2	Apex SW on lure	Portal registration
L3	Persist hooks / storage / SW cache	Pathway support
L4	Hidden `chrome://` residency via **CDP**	Shipping
L5	Isolated Web App install (Mojo)	Shipping under dev flags
L6	SW fleet start/stop (WebUI)	Shipping

L4–L6 on **upstream** Chrome do not need victim EV tabs. CDP `CreateTarget` is enough.

9.2 L6 — serviceworker-internals

Harvested page logic (`serviceworkers.js`) exposes:

```
start | stop | unregister | inspect
```

via `sendWithPromise`, plus `getAllRegistrations` and options such as `debug_on_start`. The proxy sticky-opens `chrome://serviceworker-internals/`, lists partitions, and **starts** non-running registrations whose script/scope match the apex trusted-loop SW.

L6 cannot invent a new service worker registration. It keeps existing ones alive. Re-register still needs a controlled client on the lure origin.

9.3 L5 — Isolated Web Apps (not SW internals)

IWA install is a **different WebUI** and a **different IPC style**:

```
chrome://web-app-internals
  → WebAppInternalsHandler.getRemote()   // Mojo
  → installIsolatedWebAppFromDevProxy({ url })
  → installIsolatedWebAppFromBundleUrl({ webBundleUrl, updateInfo })
```

Upstream Chrome is launched with `IsolatedWebApps` and `IsolatedWebAppDevMode`. If those flags were missing at process start, install probes report `isIwaDevModeEnabled=false` until restart. The payload must be a real IWA (dev proxy origin or signed bundle)—not an arbitrary website.

9.4 Operator maintain

```
POST /api/rbi/offscreen/maintain
POST /api/rbi/sw-internals/maintain
POST /api/rbi/iwa/install   { "url": "https://localhost:5193" }
```

Or from the portal: `maintainOffscreen()`, `maintainSWFleet()`, `iwaInstall(url)`.

10. Auth, WebAuthn, and session harvest

10.1 Auth pipelines

Specialized paths exist for Google, Okta, Microsoft, AWS, and Auth0: step detection, structured fill, settle delays, and mirror materialization tuned to each SPA. This is product work on top of the generic input relay.

10.2 WebAuthn

Upstream hooks `navigator.credentials`. Challenges flow through the proxy to the victim, who completes the ceremony at the correct RP origin; assertions return into the upstream tab so **session cookies stay upstream**. Virtual authenticator remains a lab fallback. The demo story “victim signs, middle browser wins” does not require EV magic.

10.3 Okta FastPass

A loopback shim avoids binding the proxy to `:8769` while still completing Verify-style challenges in the BiTM model.

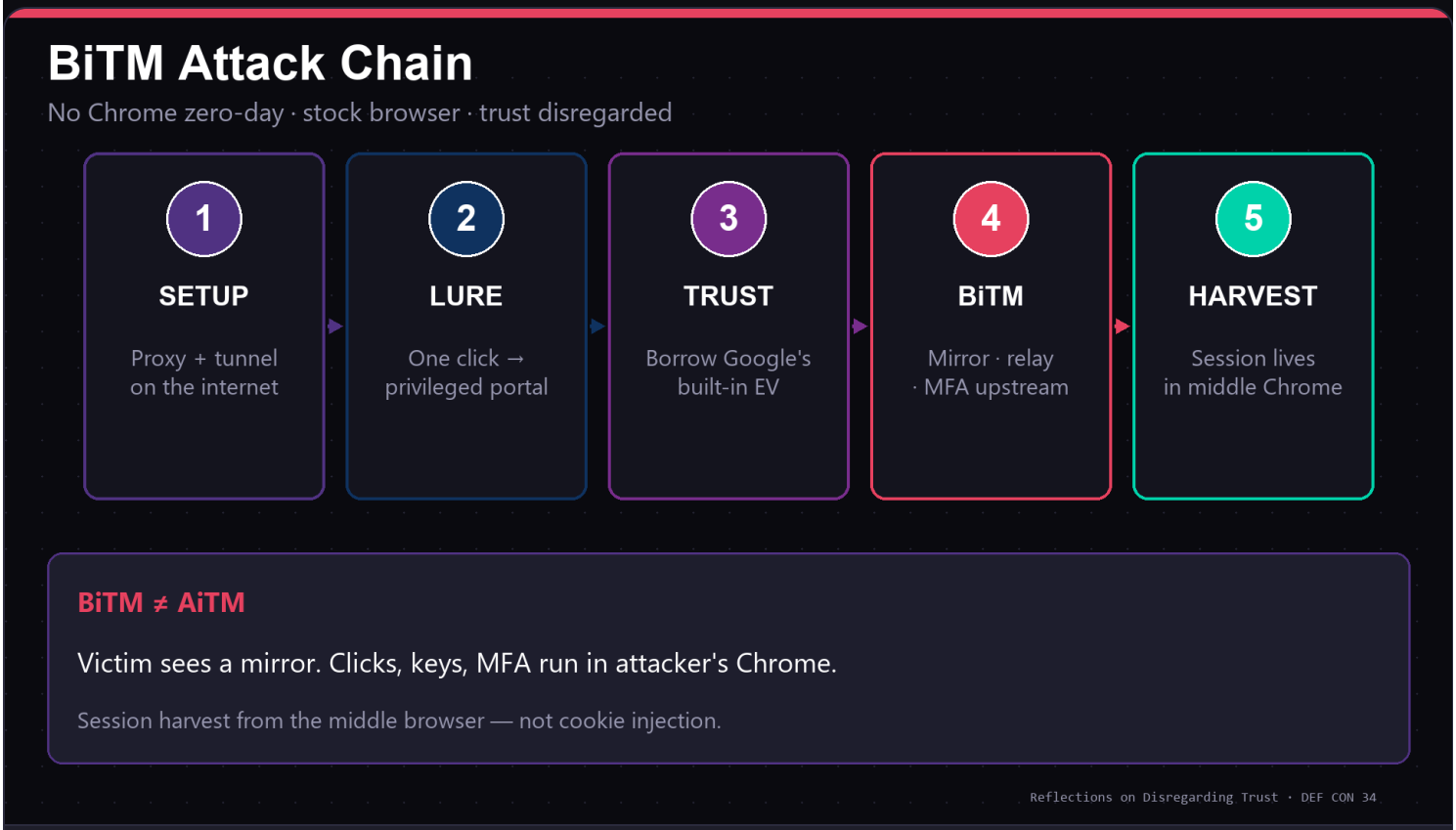
10.4 Cookies and Mojo foothold

With a chromewebdata CDP anchor, `Network.getAllCookies` and related probes expose jar material beyond ordinary web origins. EV account/sync paths add Google-oriented session visibility when magic is live. Export endpoints and client-detail inventory complete the operator feedback loop.

10.5 Production routing

`--prod` allowlists path → upstream maps. Portal paths remain the intentional exception for `?url=` streaming (see §3.1). Non-portal arbitrary targets return 403.

11. Operational chain



BiTM chain — multi-rail control, no Chrome 0-day

```
1. Deploy proxy (prefer --headed); sticky maintainer arms background agents
2. Expose via TunnelTug; set public_url
3. Lure: /dino?url=<https-idp>
4. Portal + chromewebdata foothold; apex SW registers
5. Multi-rail scan (EV primary if live; backups warm)
6. Mirror IdP; relay input
7. Complete MFA on victim device into upstream ceremony
8. Harvest cookies / inventory; optional L6 start or L5 IWA on upstream
9. If magic later fails: continue on inject + sticky rails + WebUI fleet
```

The chain is deliberately **parallelizable**: BiTM data plane and multi-rail control plane fail independently. Losing EV magic should not imply losing the middle browser session.

12. Defensive implications

12.1 Detection ideas

- WebSocket `/api/stream-sync` (or equivalent) on unexpected domains
- Shadow DOM RBI stage markers; SW script `/rbi-trusted-loop-sw.js`
- Tabs at `chrome-error://chromewebdata/` while presenting IdP UI
- Extension messages carrying magic `91556947316803` or weak host `*` from non-extension origins
- Operator Chrome profiles with parked `serviceworker-internals`, `web-app-internals`, Gemini, Meet agents
- Unexpected IWA installs or `IsolatedWebAppDevMode` on managed fleets
- Tunnel edge patterns (ACME + long-lived WSS + QUIC control)

12.2 Hardening directions

Trust issue	Direction
EV magic + host `*`	Bind RPC to true extension contexts; reject wildcard hosts from web/deputy origins
chromewebdata debugger attach	Scope cookie APIs; restrict attach on error pages
Apex `Service-Worker-Allowed: /` with extension bridges	Treat as high-risk pattern
Portal `?url=` open relay in prod	Apply egress policy uniformly to portal paths
WebAuthn through mirrors	Bind RP verification to visible origin, not presented extension origin
Component externally_connectable + documentId	Audit private APIs for confused-deputy callers
WebUI Mojo automation	Gate fleet and IWA handlers behind enterprise debug policy
Magic-only incident response	Assume multi-rail; do not declare “fixed” when only EV magic closes

12.3 Enterprise posture

Component extensions and WebUI are part of the trusted computing base of Chrome. Hardening EV alone is necessary but not sufficient. Device-bound sessions that cannot be completed into a foreign browser, plus monitoring for RBI-shaped mirrors and deputy contexts, address the BiTM class more directly than “block unknown CRX.”

13. Implementation map

Core Go packages (`internal/app/`)

Area	Examples
Process / mux / CDP launch	`proxy.go` (incl. IWA feature flags)
Mirror / session / input	`rbi_sync.go` , `rbi_session.go` , `input_relay.go` , `stream_mirror.go`
Foothold	`trusted_loop.go` , `chromewebdata_agent.go` , `chrome_mirror.go`
Sticky control	`sticky_agents.go` , `glic_relay.go` , `component_relay.go`
L5 / L6 WebUI	`iwa_internals.go` , `sw_internals.go`
MFA / auth	`webauthn_relay.go` , `okta_fastpass.go` , `auth_*.go`
EV / probes	`ev_rpc.go` , `client_details.go` , `downstream_rpc.go`

Portal assets (`src/assets/dino/`)

Area	Examples
Multi-rail	`backup_foothold.js` , `glic_foothold.js`
EV / deputy	`ev_bridge.js` , `endpoint_validation_client.js` , `offscreen_script.js` , `ev_origin_present.js`
SW / inventory	`trusted-loop-sw.js` , `trusted-loop-inventory.js` , `trusted-loop-probe.js`
Persistence	`persistence_pathway.js` , `webui_foothold.js`

Research tooling

Dagger, Loki, Helheim, and pak extraction under `tools/loki-closure/` turn first-party packs into surface maps that feed the live rails above.

Operator docs in-repo

Doc	Topic
`docs/backup-foothold.md`	Multi-rail runbook
`docs/iwa-install.md`	L5 IWA install
`docs/trusted-loop.md`	Dino / EV / SW
`docs/ev-foothold-webauthn-relay.md`	Demo: foothold + DOM/key + victim-sign
`docs/architecture.md`	System map

14. Disclaimer

This document describes mechanisms present in the EvilRBI / `~/proxy` codebase for **authorized security research, red teaming, and defensive analysis**. Use against systems or people without explicit permission is unlawful and unethical. The MIT license does not authorize misuse.

Closing

EvilRBI is advanced not because it invents a new class of memory corruption, but because it **composes the browser's own control surfaces** into a resilient BiTM system: middle-browser authenticity, deputy foothold, component rails, sticky residency, and WebUI fleet and IWA paths—with EV magic as a powerful primary that operators can use today while parachutes stay packed.

No 0-days. Stock Chrome. Disregarded trust.

Reflections on Disregarding Trust · DEF CON 34 · `~/DC34/EvilRBI-Whitepaper.md`
Professional narrative revision: multi-rail control plane, sticky agents, L4–L6 WebUI, L5 IWA; EV primary with continuity preplanning.