From Prompt to Phish: How AI-Generated Tooling Is Industrializing Modern Identity Compromise

Author: Piotr Mazurkiewicz
No of pages: 78
Read time: 90 minutes

Contents

1 Executive Summary

This report examines an emerging ecosystem of **vibe-coded OAuth phishing and Microsoft 365 account takeover platforms** that industrialize the full lifecycle of Business Email Compromise operations. The analyzed tooling shows how attackers are moving beyond traditional credential theft toward **token-centric compromise**, using OAuth Device Code phishing, refresh token import, PRT and reusable session-cookie generation, mailbox intelligence collection, and automated downstream exploitation.

The core finding is that **DC Broker**, **DC-Inbox**, **BoB V2**, and **BoB V3** appear to form a **connected criminal tooling ecosystem** rather than isolated utilities. DC Broker focuses on acquisition and token management; DC-Inbox converts stolen Microsoft 365 access into mailbox intelligence and impersonation capability; BoB V2 operationalizes mass contact harvesting, inbox rule abuse, and email sending; and BoB V3 advances the model further with modular architecture, AI-assisted enrichment, multi-account campaign automation, bot-gate protection, and multi-tenant API key management.

The platforms demonstrate a **high level of operational maturity**. They combine professional user interfaces, multi-operator workflows, Telegram notifications, proxy and VPS provisioning, mailbox caching, **AI-driven analysis**, and automated token forwarding into an integrated pipeline. This allows a threat actor to progress from initial OAuth token capture to mailbox surveillance, contact harvesting, phishing infrastructure deployment, and fraudulent email delivery with minimal manual effort.

From a defensive perspective, the most important implication is that Microsoft 365 compromise must be treated as a **token, session, and behavior problem** rather than only a password or MFA problem. Device Code Flow abuse, refresh-token reuse, PRT-style session material, malicious inbox rules, proxied Graph access, and suspicious mailbox automation all require explicit monitoring, conditional access controls, session revocation processes, and detection engineering.

The **business risk is significant** because these platforms directly support BEC, internal spear-phishing, vendor impersonation, payment diversion, data exfiltration, and persistent mailbox monitoring. Their use of legitimate Microsoft authentication endpoints, genuine compromised accounts, geographically aligned proxies, and mailbox-native sending mechanisms reduces the effectiveness of controls that focus only on malicious domains or spoofed email infrastructure.

Priority mitigations include restricting or monitoring OAuth Device Code Flow usage, enforcing phishing-resistant and context-aware access policies, enabling Continuous Access Evaluation, monitoring anomalous Graph API and mailbox access, detecting new or modified inbox rules, reviewing risky OAuth consent and token activity, and building incident response playbooks that include token revocation, session invalidation, mailbox rule cleanup, and post-compromise email review.

Overall, the analyzed ecosystem represents a **mature, service-like approach** to Microsoft 365 abuse: acquisition, intelligence, automation, infrastructure, and monetization are increasingly integrated into a single operational chain. **Organizations should assume that successful token theft can rapidly escalate into persistent mailbox access and trusted**

> **channel fraud**, and should prioritize controls that detect and disrupt the entire lifecycle rather than only the initial phishing event.

Relationship between BoB V2/V3 and DC Broker

BoB V2 and its successor BoB V3 are tightly integrated with DC Broker via the RetroBoB push-token mechanism: when DC Broker captures a new token, it can automatically POST it to BoB V2/V3's POST /api/retrobob/push-token endpoint, making the account immediately available in BoB without any manual operator action. This creates a fully automated pipeline from phishing to exploitation.

Key points of the paper:

This research is intended to provide in-depth analysis of the phishing panels discovered, however as this may be time consuming to read through the whole file, the most important information can be found in those key sections:

1.1 The Rise of OAuth-Based Account Takeover

The widespread adoption of Microsoft 365 (M365) in enterprise environments has made it a primary target for credential theft operations. Traditional password-based phishing has been progressively displaced by techniques that bypass multi-factor authentication (MFA) entirely by targeting the OAuth 2.0 token layer rather than the password layer.
Two primary approaches dominate modern M365 account takeover:

- Adversary-in-the-Middle (AiTM): Reverse proxy frameworks (Evilginx, Modlishka, Muraena) intercept the full authentication session, capturing both the user's session cookie and the MFA response in real time. Tools like Evilginx2 and its successors operate in this category.
- Device Code Flow Phishing: The attacker initiates an OAuth Device Code authentication request, obtains a user code, and tricks the victim into entering the code at Microsoft's legitimate microsoft.com/devicelogin page. The victim completes MFA on a legitimate Microsoft page; the attacker's polling loop receives a valid refresh token. This technique is documented in Microsoft security advisory and is particularly effective because no phishing proxy is required — the victim interacts only with genuine Microsoft infrastructure.

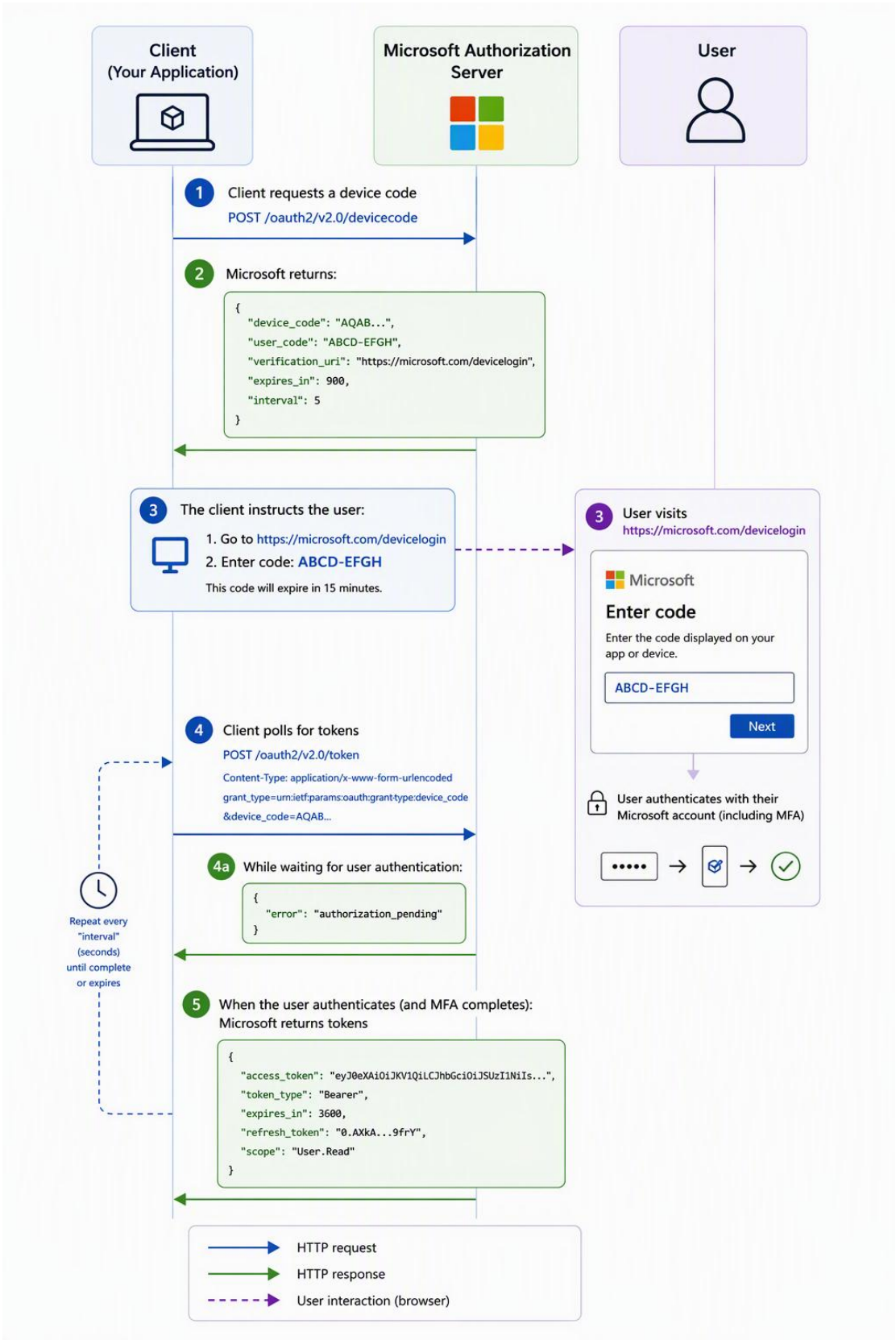
DC Broker industrializes the second technique while also providing infrastructure to operationalize tokens obtained by either method, including direct import of refresh tokens harvested by external AiTM frameworks.

Business Email Compromise Context

Stolen M365 refresh tokens and PRT cookies are the entry point to high-value BEC operations, enabling attackers to conduct internal spear-phishing from hijacked executive accounts, manipulate invoices for payment fraud, move laterally into SharePoint, Teams, OneDrive, and Azure, exfiltrate data from mailboxes and cloud storage, and impersonate trusted vendors or employees. DC Broker explicitly acknowledges and facilitates these downstream uses through its mailbox reading, contact enumeration, inbox rule management, and email sending capabilities.

1.2 Device Code Flow

The OAuth 2.0 Device Authorization Grant (RFC 8628) was designed for input-constrained devices (smart TVs, IoT devices) that cannot display a browser. The flow is:


Client
(Your Application)
Microsoft Authorization Server
User
1
Client requests a device code
POST /oauth2/v2.0/devicecode
2
Microsoft returns:
{
"device_code": "AQAB...",
"user_code": "ABCD-EFGH",
"verification_uri": "https://microsoft.com/devicelogin",
"expires_in": 900,
"interval": 5
}
3
The client instructs the user:
1. Go to https://microsoft.com/devicelogin
2. Enter code: ABCD-EFGH
This code will expire in 15 minutes.
3
User visits
https://microsoft.com/devicelogin
Microsoft
Enter code
Enter the code displayed on your app or device.
ABCD-EFGH
Next
User authenticates with their Microsoft account (including MFA)
4
Client polls for tokens
POST /oauth2/v2.0/token
Content-Type: application/x-www-form-urlencoded
grant_type=urn:ietf:params:oauth:grant-type:device_code
&device_code=AQAB...
4a
While waiting for user authentication:
{
"error": "authorization_pending"
}
Repeat every "interval" (seconds) until complete or expires
5
When the user authenticates (and MFA completes):
Microsoft returns tokens
{
"access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIs...",
"token_type": "Bearer",
"expires_in": 3600,
"refresh_token": "0.AXkA...9frY",
"scope": "User.Read"
}
HTTP request
HTTP response
User interaction (browser)

Because the user authenticates against the real Microsoft endpoint, no phishing proxy is required, MFA is completed legitimately, and no certificate warnings are shown. This technique defeats: - Phishing-resistant MFA (FIDO2/passkeys) except in some configurations - Conditional Access policies that check sign-in location if the attacker uses a matching proxy - Anti-phishing browser warnings.

1.3 Shikamaru – a mysterious AI component

Shikamaru appears to be a custom-built or privately branded AI component rather than a publicly documented product, framework, or open-source model. Its name surfaces inside the tooling as an internal service layer that exposes dedicated API routes for mailbox analysis, intelligence item generation, timeline construction, vendor and relationship mapping, and deeper research over selected email threads. The use of an API key format consistent with OpenAI-compatible providers suggests that Shikamaru may not be a standalone model itself, but rather an orchestration wrapper around one or more commercial or self-hosted LLM backends, tailored specifically for post-compromise mailbox intelligence. In practical terms, it should be treated as an attacker-side AI analyst: a custom component designed to turn large volumes of stolen email into prioritized, actionable intelligence for BEC, fraud, and follow-on phishing operations.

1.4 Vibe-coded tools

All tools analyzed in this report appear to have been developed with AI or LLM assistance, and the confidence level for this assessment is extremely high. The evidence supporting this conclusion includes following:

1.4.1 DC Broker Dashboard

Extremely consistent decorative comment style

DC Broker contains 132 instances of box-drawing separator comments:

```
// ═══════════════════════════════════════════════════════════
// Theme
// ═══════════════════════════════════════════════════════════

/* ──── Stale-While-Revalidate "updating…" pill ──── */

/* ────────── Login screen ────────── */

/* ── 3-pane layout ──────────────────────────────────── */
```

This decorative, perfectly aligned comment style is a strong AI generation signal. Human developers writing code over time develop inconsistent, pragmatic comment styles. AI models consistently produce this exact ═══ and ── border style when

generating structured code sections. The density (132 occurrences in one file) and consistency of the pattern is not characteristic of organic human development.

Backup File with Timestamped AI Redesign Reference

`templates/dashboard.html.bak.outlook-redesign-2026-04-28`

This is explicit evidence of an AI-assisted redesign session. The pattern — save a backup, run an AI session to redesign a specific component, produce a new version. It is the standard vibe-coding workflow. The date `2026-04-28` shows the Outlook-style inbox was a specific, dated AI generation task.

CSS Variable Over-Engineering

563 instances of `var(--...)` CSS custom properties in a single file, with a complete dual-theme (light/dark) design system and 60+ named variables. This level of CSS architecture is consistent with AI generation — models are trained on design system documentation and tend to produce complete, symmetrical token sets when asked to build a polished UI. A human developer building a criminal tool would not invest this level of CSS token discipline.

Async/Await Density

345 `async function` / `await` patterns in a single file. While this is appropriate for an API-heavy application, the absolute uniformity of the pattern (no callbacks, no `.then()` chains, no mixed patterns) is consistent with AI generation within a single or small number of sessions. Human codebases accumulate mixed async patterns over time as different developers and different JavaScript versions are involved.

1.4.2 DC-Inbox

Explicit Chunk Development Manifest

The most direct evidence of vibe coding in the entire codebase is this comment block in dc-inbox.html:

```
/*  Original code authored for the dc-broker app. Calls existing /api/*

    endpoints — no backend changes required. Modules:

      • Folders + List + SSE delta (chunk 2)

      • Reader + attachments + download .eml (chunk 2)

      • Search bar (chunk 2)

      • Import drawer (.json + .js cookies) (chunk 3)
```

```
    • Tags sidebar + tag-from-reader (chunk 4)
    • Monitored keywords sidebar + add/edit/delete + hits (chunk 4)
    • Compose drawer (new/reply/forward, multipart send) (chunk 4)
    • Insights right rail (sync, kw stats, recent hits) (chunk 4)
*/
```

This is a literal AI session planning document embedded in the source code. The term "chunk" is the exact language used when prompting AI code generators to produce large outputs in segments: "Generate chunk 1 of this application… now generate chunk 2…". Each "chunk" corresponds to one or more AI generation sessions. The manifest serves as a tracking document for what has been generated and what remains — a workflow management artefact of AI-assisted development.

Supporting evidence: the HTML contains disabled UI elements explicitly labelled as pending chunks:

```
<button class="btn-compose" disabled title="Available in Chunk 4">
<button title="Add keyword (Chunk 4)" disabled style="opacity:.4">+
</button>
```

These disabled buttons are stubs placed by the AI during an earlier chunk generation, marking where the next chunk would attach. This is a standard AI vibe-coding pattern: the AI generates a complete-looking UI shell with placeholder disabled elements, then fills them in progressively across subsequent prompts.

data-chunk Injection Markers

The opening comment also documents the approach:

"Chunks 2–4 will progressively wire list → reader → import → right rail + monitored keywords."

This is the verbatim language of iterative AI-assisted development: "wire" is the characteristic AI vocabulary for connecting UI components to their backend calls.

132-Line CSS Variable System Matching DC Broker's Pattern

DC-Inbox replicates the same dense CSS custom property architecture as DC Broker — 421 var(--...) instances using the same token naming convention. The two codebases clearly came from the same AI prompt lineage: "Build an Outlook-style inbox that matches the design system in the existing dashboard."

1.4.3 BoB V2

Single-File Architecture with Explained Design Decisions

BoB V2 contains unusual inline comments that explain the *why* behind CSS decisions. It's a pattern AI models produce when generating code with rationale:

```
/* Three layered cues so VPS-induced latency never feels frozen:
 *   1. #topLoader  — a thin rainbow/accent bar …
 *   2. .is-busy    — auto-applied by the api() wrapper to the button
 *   3. .shimmer    — shimmering placeholders instead of stale numbers
 */
```

This self-documenting style — justifying implementation choices inline — is more characteristic of AI output than human developer notes. Human developers write *what* the code does; AI models explain *why* they made architectural choices, because they are trained to produce explanatory documentation.

Box-Drawing Comment Style (77 instances)

BoB V2 contains 77 instances of the same `// ———` / `/* ═══ */` separator style seen in DC Broker. The lower count (vs 132 in DC Broker) is consistent with BoB V2 being an earlier, slightly less AI-polished tool.

API Surface Documentation in CSS Comments

The CSS comments reference specific API endpoints by name:

```
/* ── Auth: login overlay, impersonation banner, user menu, admin modal ── */
```

Embedding API/function names in CSS section comments is AI behavior for when generating a large frontend file, the AI uses section headers that reference the full application feature set for coherence. A human developer would typically not write CSS comments referencing backend concepts.

More Human Characteristics

BoB V2 also exhibits characteristics more consistent with human authorship: Fewer `async/await` patterns (51 vs 345 in DC Broker), More inline style attributes (153 vs 417 in DC Broker), suggesting less architectural discipline, possibly more manual iteration, The escaping functions (`esc()`) are called less systematically (43 vs 145 in DC Broker), suggesting a human reviewer added them reactively rather than an AI generating them comprehensively from the start

Assessment: BoB V2 appears to be a hybrid of AI-generated structure and boilerplate with human modification and iteration on top. Possibly generated by an earlier, less capable AI model, or generated with less complete prompts.

1.4.4 BoB V3

API Documentation at Module Headers

Every JavaScript module in BoB V3 begins with a meticulously formatted API documentation block:

```
/* BoB V3 — compose + send (chunk 3)
 *
 *   GET  /api/account/{id}/settings
 *   PUT  /api/account/{id}/settings
 *   POST /api/account/{id}/send-test
 *   POST /api/account/{id}/send-individual
 *   POST /api/account/{id}/send-bcc
 *   POST /api/account/{id}/change-sender-name
 *   GET  /api/account/{id}/contacts
 *   GET  /api/contacts/all
 *   GET  /api/contacts/third-party
 *   ... (29 total)
```

This practice — listing every API endpoint a module uses at its top — is not a standard human development convention. It is an AI generation artefact: when prompted to generate a module, the AI produces a complete header that documents its API surface (because it has been trained on well-documented open source code). The consistency of this pattern across all 9 modules confirms they were generated by the same AI workflow.

`data-v3-injected="1"` Attribute

Multiple `<main>` sections in `BoB V3.html` carry a `data-v3-injected="1"` attribute:

```
<main class="main view" id="view-send" data-view="send" hidden="" data-v3-injecte
d="1">
<main class="main view" id="view-pro" data-view="pro" hidden="" data-v3-injected=
"1">
```

This attribute has no functional CSS or JavaScript usage — it is purely a **metadata marker**. The pattern `data-v3-injected` strongly suggests these sections were programmatically inserted into the HTML shell by an automated process — either an AI that marked its own generated sections, or a build script that injected AI-generated HTML blocks into an existing shell. Either way, it is direct evidence of AI-assisted generation with explicit provenance tracking.

Refactoring from Monolith to Modules - The well-known AI Refactor Pattern

The transition from BoB V2 (6,091-line monolith) to BoB V3 (2,433-line shell + modules) follows the exact pattern of AI-driven refactoring: *"Here is my existing single-file application. Refactor it into a modular structure where each view has its own*

JavaScript file." The AI produces a shell HTML file with mount points and separate JS files for each module — exactly what BoB V3 is.

The `const api = (...a) => window.api(...a)` pattern in every module (delegating to a global `api()` function defined in `core.js`) is a textbook AI-generated modularization pattern — it provides module isolation while maintaining access to the shared `api()` wrapper without imports, because the AI knows this file will be loaded in a browser context where module bundling is not available.

`retrobob.js` — Self-Describing Pipeline Comment

```
/* Automated token pipeline — upload / push-token → analyze → extract → send */
```

This single-line description of a four-stage automated pipeline is characteristic of AI-generated summary comments — the AI provides a high-level natural language description of what it just generated, which happens to be exactly the kind of comment that would appear in an AI chat response: *"Here is the RetroBoB module. It implements an automated token pipeline: upload / push-token → analyze → extract → send."*

1.4.5 Shared Design Language between the tools

All four tools use the same CSS custom property naming conventions (`--bg-card`, `--text-muted`, `--accent`, `--border`), the same box-drawing comment separators, and the same async/await API wrapper pattern. This consistency across tools suggests either:

- A single developer used the same AI system across all four tools, or
- The AI was prompted with previous tool code as context, producing stylistically consistent output

In either case, the stylistic consistency is more consistent with AI generation (which produces stylistically uniform output within a training paradigm) than with a human developer team (which accumulates stylistic divergence over time).

2 Attack Chain examples

The following example illustrates how the observed tooling can be chained into a complete Microsoft 365 account takeover and exploitation workflow, starting with infrastructure preparation and victim delivery, then progressing through token acquisition, session access, mailbox intelligence collection, BEC execution, and persistence. It demonstrates that the individual capabilities described throughout this report are not isolated features, but components of an integrated operational pipeline that can move from initial OAuth abuse to trusted-channel fraud with limited manual effort.

Phase	Operational flow
1. Infrastructure Setup	Deploy phishing page to FTP domain → configure Telegram notification bot → configure RetroBoB forwarding endpoint → align proxy countries with target geography
2. Victim Delivery	Generate device code → craft lure via email, Teams, SMS, or LinkedIn → deliver verification URI and user code → wait for victim authentication through the polling loop
3. Token Acquisition	Receive refresh token after MFA completion → save token in DC Broker → trigger Telegram capture notification → forward token to RetroBoB if configured → start background EML sync
4. Account Access	Generate PRT cookie from refresh token → inject cookie into browser → access victim Microsoft 365 session → pivot into SharePoint, Teams, OneDrive, and Azure
5. Intelligence Extraction	Read mailbox and live-fetch latest emails → enumerate contacts → scan for finance and credential keywords → run Shikamaru AI analysis → identify finance staff, approvers, and high-value targets
6. BEC / Fraud Execution	Compose or reply as the victim → hijack active payment threads → create inbox rules to hide alerts or forward mail → phish trusted contacts → move funds or exfiltrate data
7. Persistence	Reuse nSSO cookie for extended access → retain downloaded EML archive after token expiry → continue keyword monitoring on active tokens → re-import or re-phish before expiry

3 Phishing attack

The phishing kits were identified during the real case investigation done by Atos SOC for a customer alert related to potentially malicious URL click. Initial sandbox analysis indicated behavior didn't match any previously observed phishing campaigns.

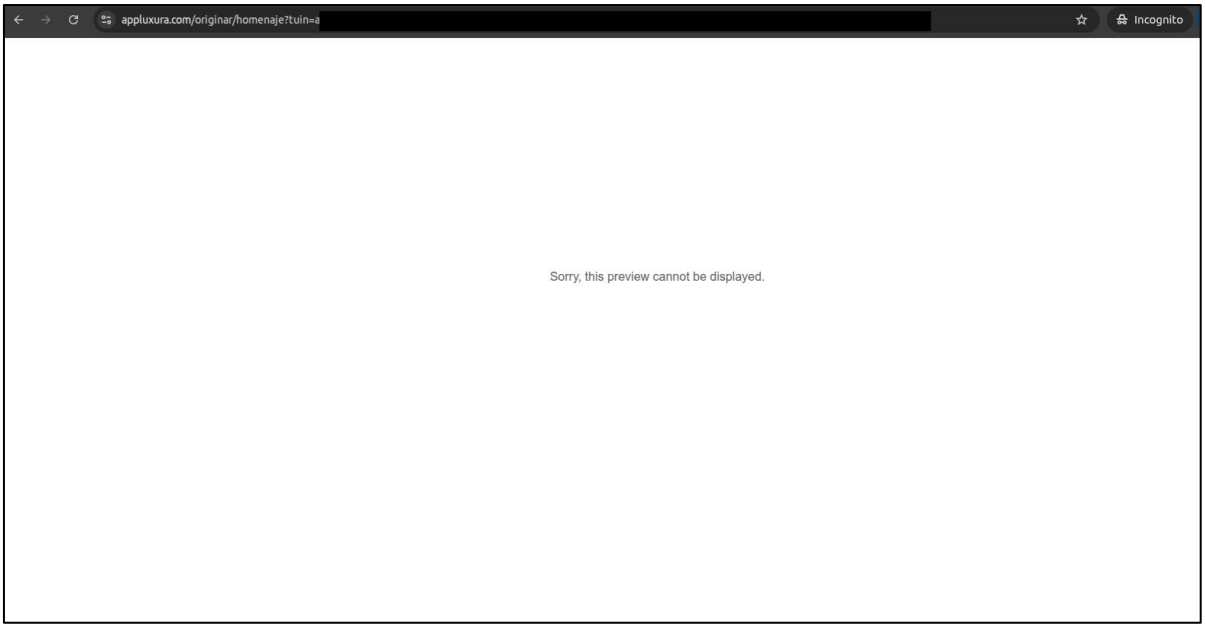


Figure 1 Visiting malicious sites with VPN shows generic errors.

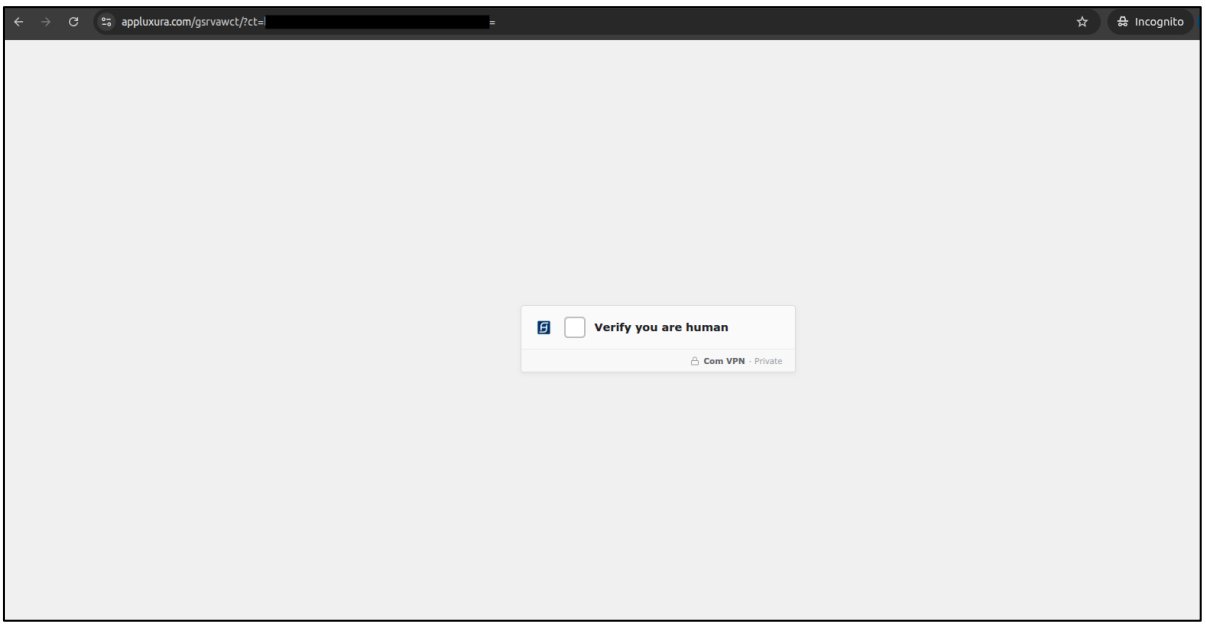


Figure 2 The same webpage through different IP address.

As usual, the URL contains a base64 encoded email address of the victim which is later passed as a variable.
Additionally, this campaign incorporates some measurements to block VPNs and well-known sandboxes from accessing the phishing pages.

The dynamic characteristic of this method is one of the reasons why it's outstanding even in today's methods. It's important to note that the phishing link was delivered to the Victim via email.

3.1 Device Code landing pages

Below are few examples of various phishing techniques created via the kit described further:

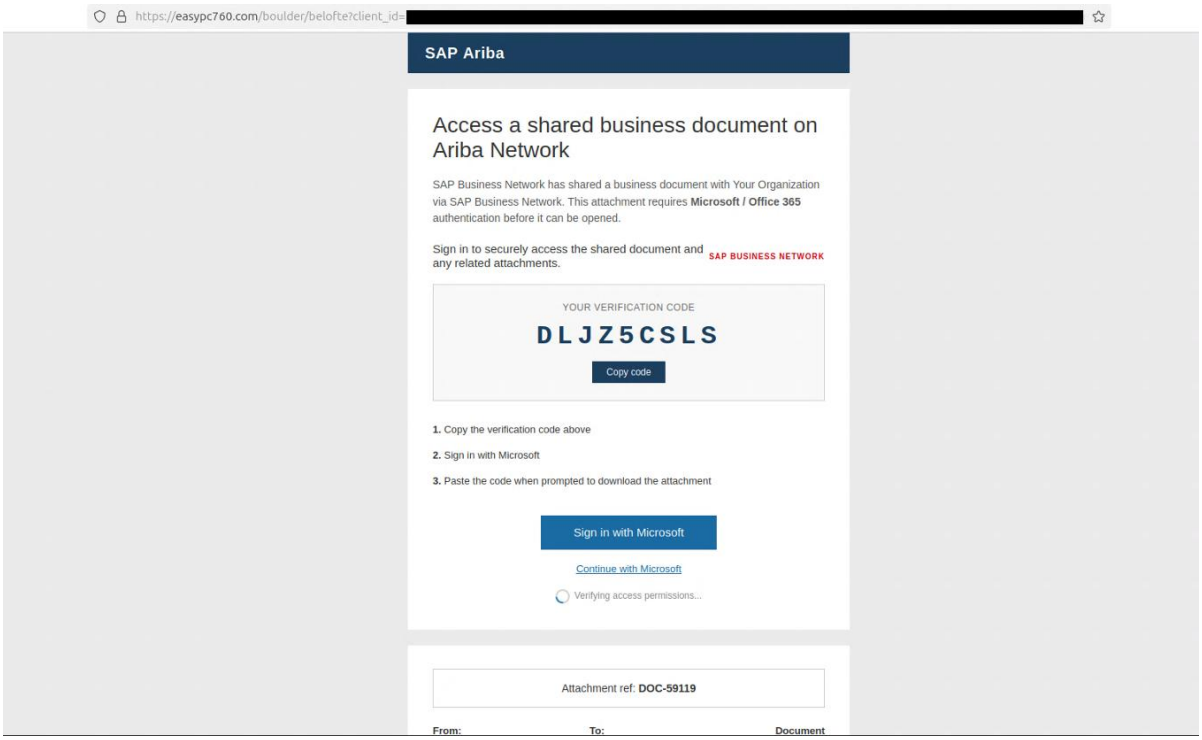


Figure 3 Device Code phishing attempt mimicking SAP Ariba access

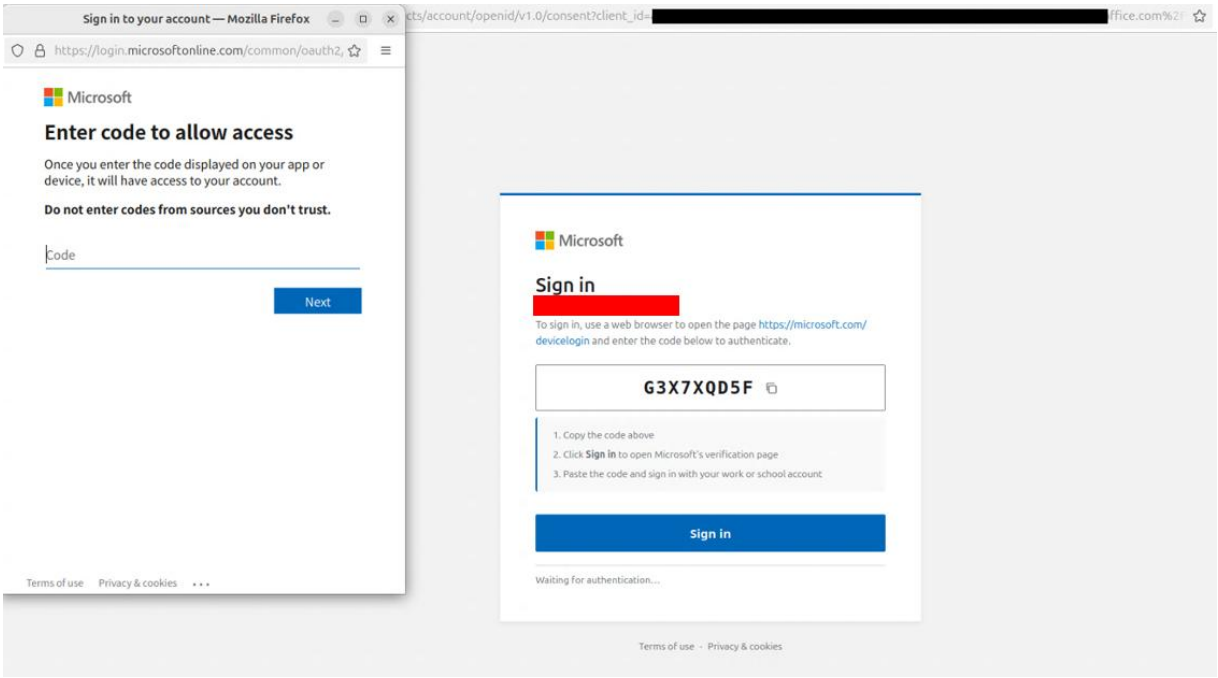


Figure 4 Another example of Device Code phishing attempt

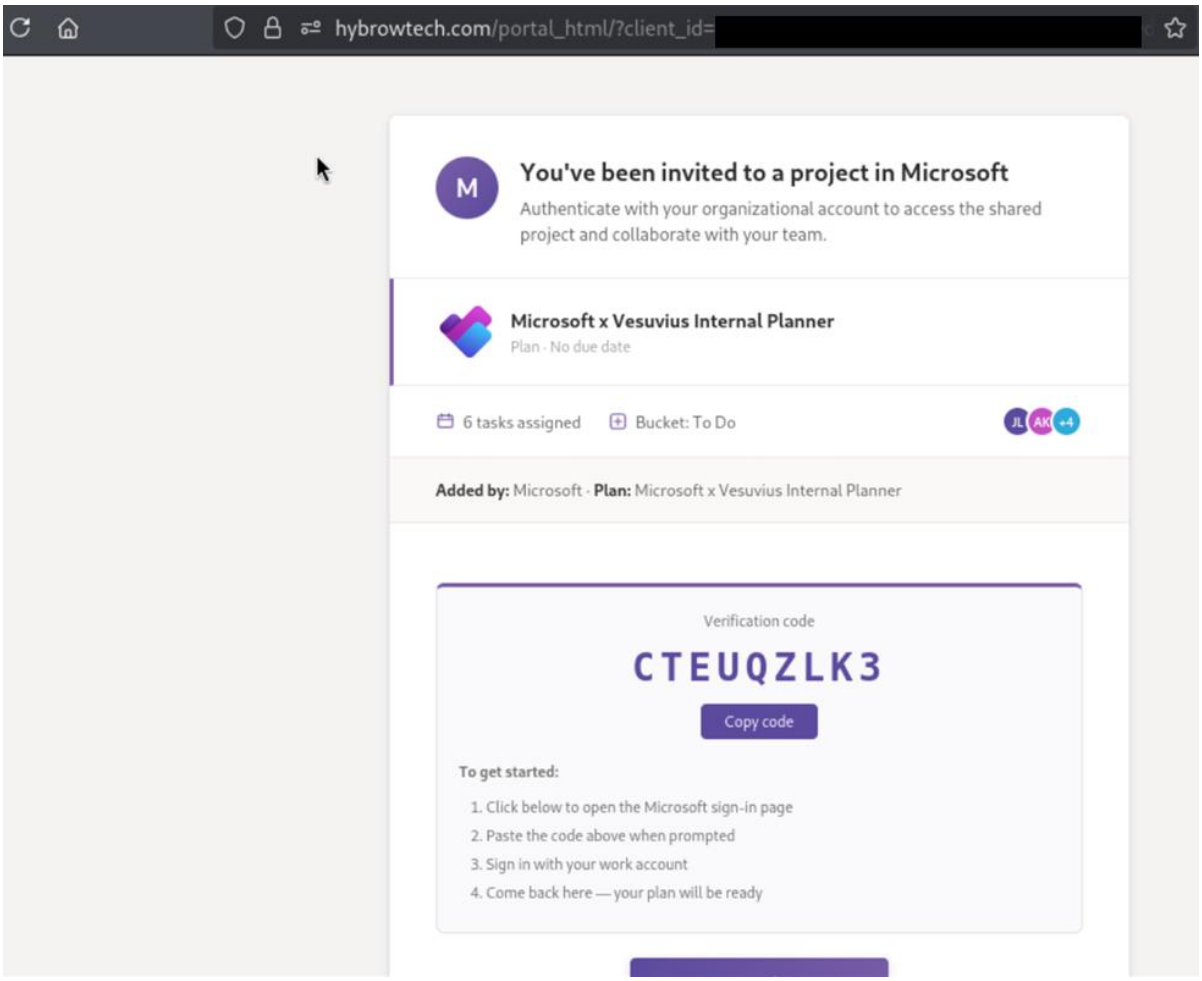


Figure 5 Another example of Device Code phishing attempt

3.2 Device Code source code analysis

Analysis of the source code available in the browser Developer Options provide a bit more information about the app.

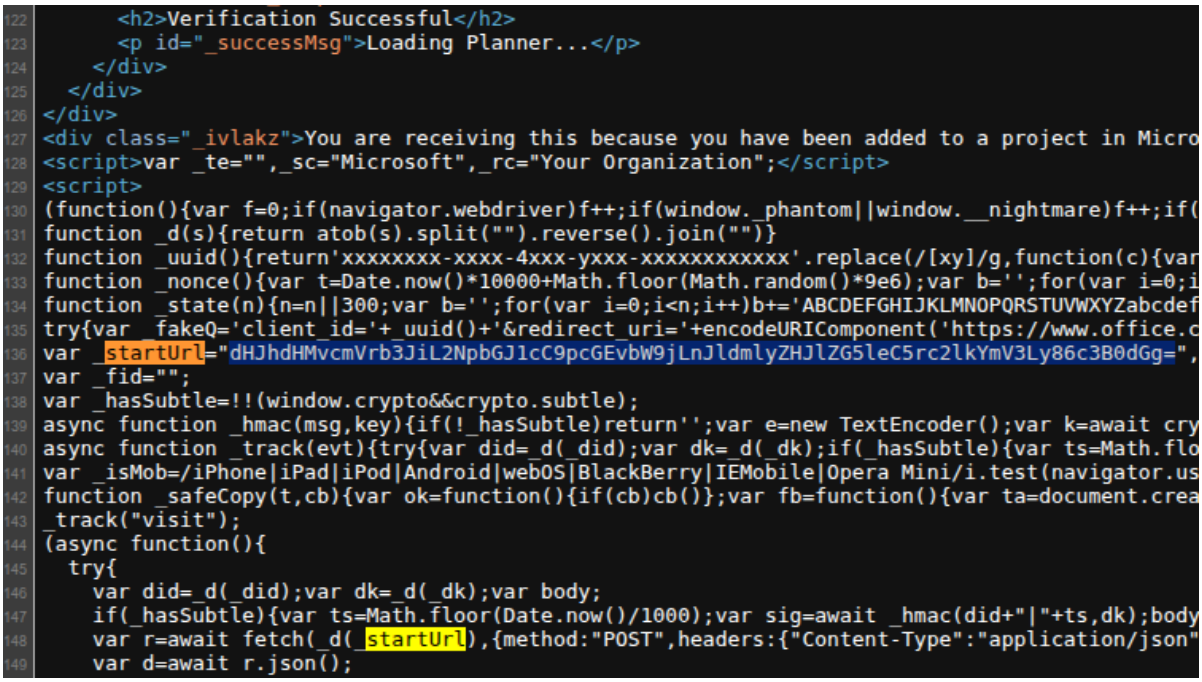


Figure 6 Source code of phishing page with default configuration

The `_te` variable is used to store victims email address, and `_rc` is used to store victim's company name. The most interesting variable is `_startUrl` which stores a reversed and base64 encoded address of the phishing kit administrative panel.

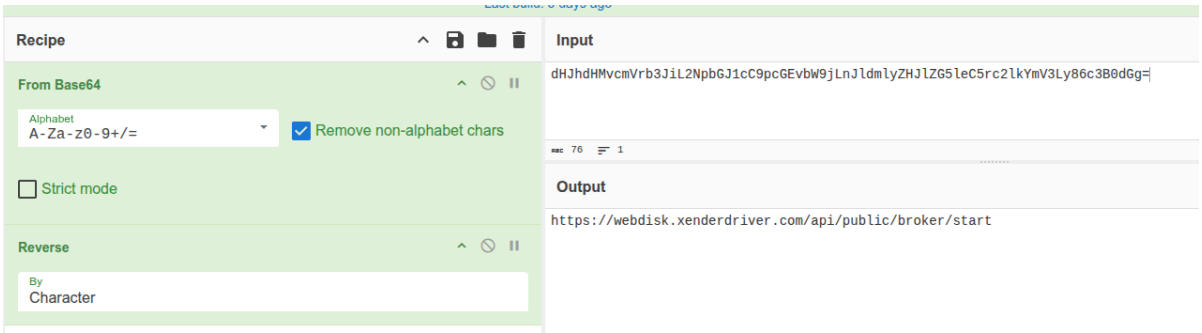


Figure 7 Recovering phishing kit address via CyberChef recipe

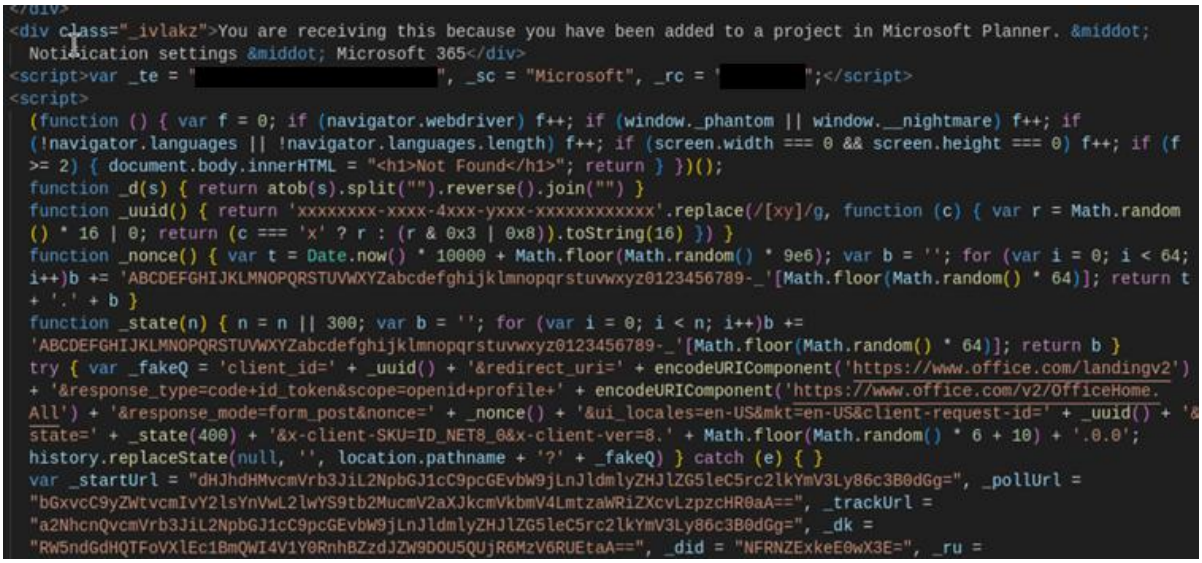


Figure 8 Source code of phishing page with victim details (blanked)

3.3 Discovery of the websites serving the phishing

The variable (URL) recovered from the source code of the webpage is a phishing kit administration portal which was available on the moment of this research at a URL of `https[:]//webdisk.xenderdriver[.]com/`

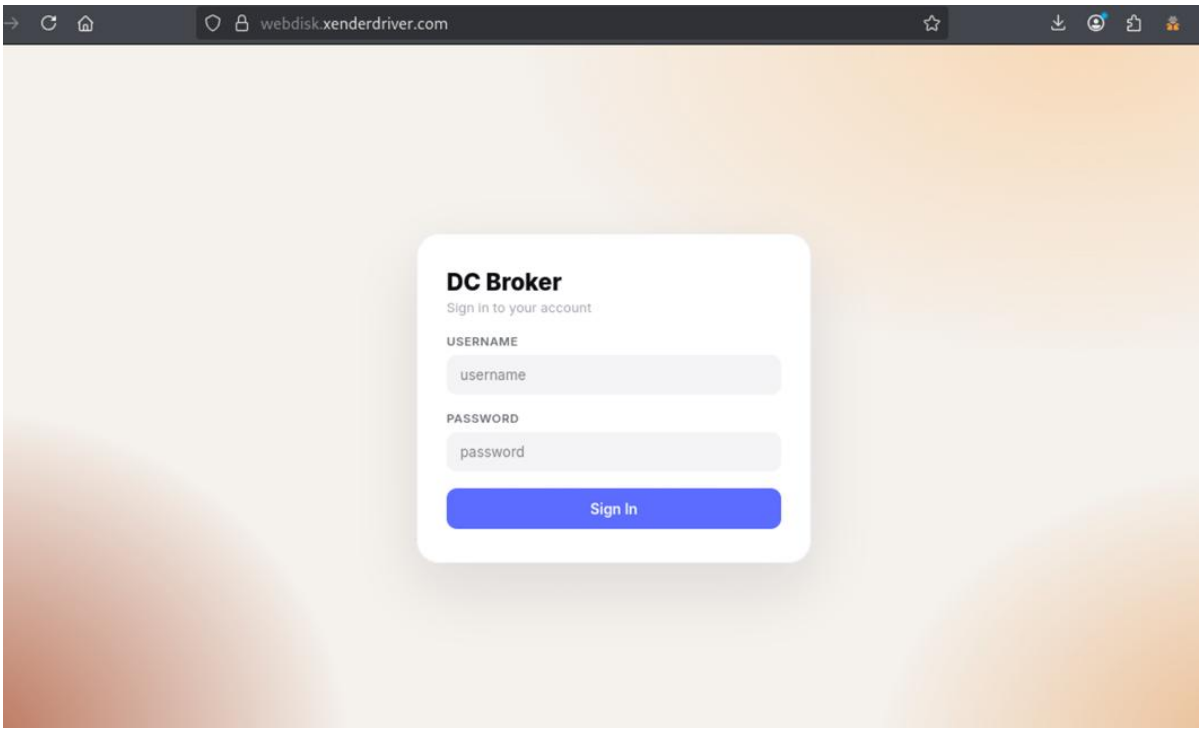


Figure 9 Login page of DC Broker phishing kit

In the classic scenario the content of the portal would stay a mystery to the researcher, however quick look at a source code reveals a bit more information about technology that was used to prepare the dashboard.

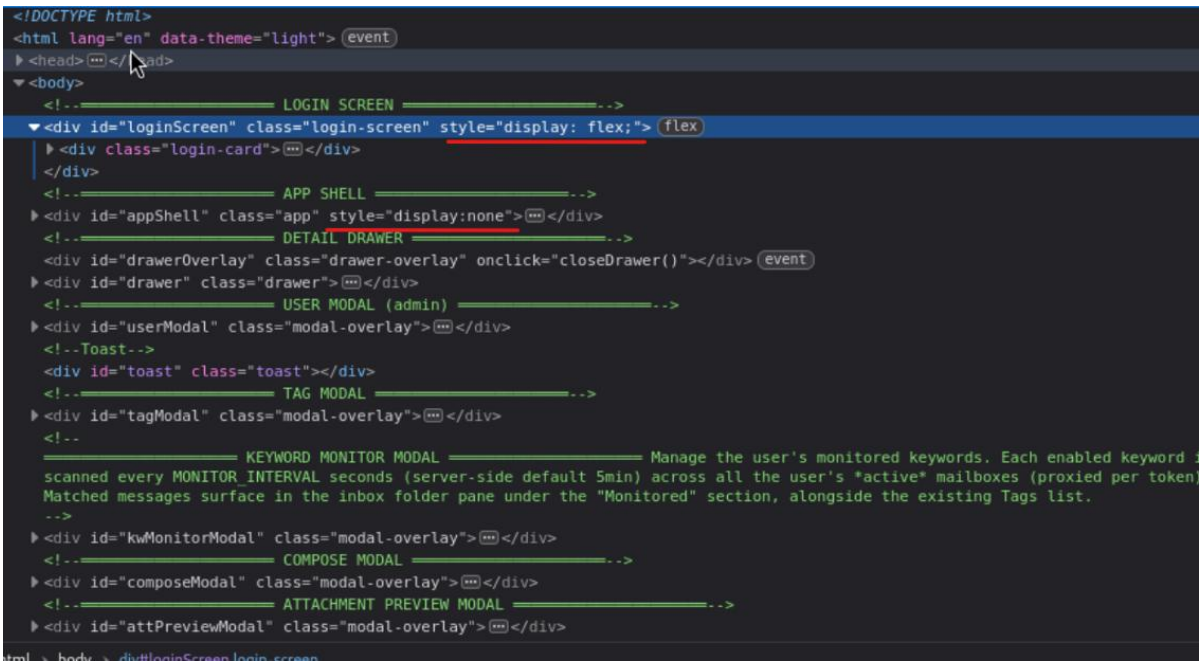


Figure 10 Interesting CSS variables that are bluring the dashboard behind login screen

The backend is clearly a Flask app, and frontend is prepared in HTML and JavaScript mix. The functionality of the app can be easily discovered due to the fact that whole dashboard frontend is being loaded at visit. Small CSS modification via Developer Options allows to hide log-in prompt, change the appShell to be shown, and modify user role to be "admin" to see more.

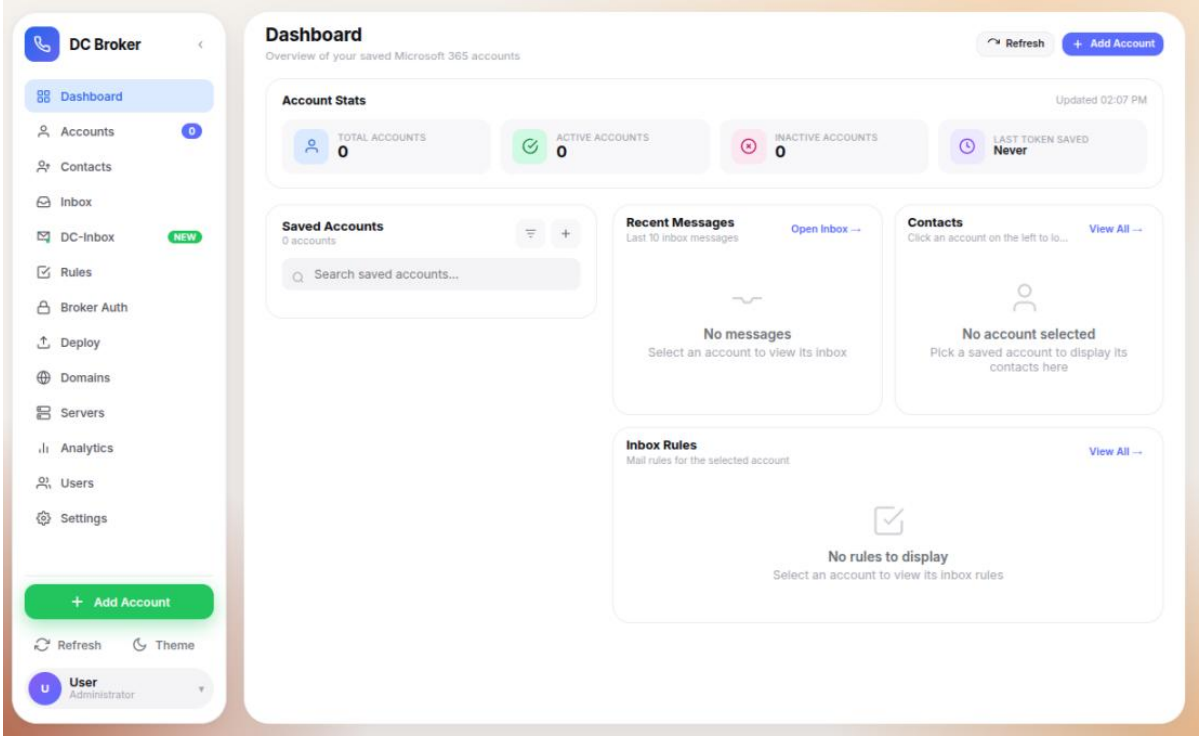


Figure 11 DC Broker Dashboard view after "bypassing" login prompt

The settings, phishing sites deployed and victims' data won't load due to the backend requiring proper authentication, however the functionality of the app can be discovered and described. **For the purposes of this research, we assumed that every visible feature in the app is fully functional.**

The same Developer Options/Source Code modification approach has been used to analyze other applications, as they appear to be related to each other, the same frontend mechanism is shared.

4 DC Broker Dashboard - An Adversarial Microsoft 365 Account Takeover Platform

"DC Broker Dashboard" is a purpose-built, multi-operator web panel designed to automate the acquisition, management, and exploitation of stolen Microsoft 365 OAuth credentials. The platform provides a unified interface for conducting OAuth Device Code phishing campaigns, managing harvested refresh tokens at scale, generating Primary Refresh Token (PRT) session cookies, reading victim mailboxes in real time, deploying phishing infrastructure via FTP, and forwarding captured tokens to downstream criminal platforms. This research performs a full technical dissection of the platform's architecture, operational workflows, and threat model, and situates it within the broader landscape of Adversary-in-the-Middle (AiTM) and Business Email Compromise (BEC) attack chains.

DC Broker is a HTML single-page application (index.html, approximately 8,900 lines) that acts as the complete frontend for a Flask-based backend. According to the data acquired it is believed that technical stack is Flask for backend operations and HTML + JavaScript for Frontend, everything working through rEST api. The panel supports multitenancy and is prepared to be quickly deployed.

Multi-Operator Model

The platform is explicitly designed for multiple simultaneous operators. Each user has:

- Their own isolated token pool
- Configurable proxy country assignment (for per-operator geolocation spoofing)
- Individual Telegram notification channels
- Individual RetroBoB API keys

Accounts (tokens) are owned per-user but can be viewed in aggregate by admins. This architecture suggests DC Broker is operated as a shared criminal service, either a private team operation or a commercial crimeware-as-a-service (CaaS) platform.

4.1 Application overview

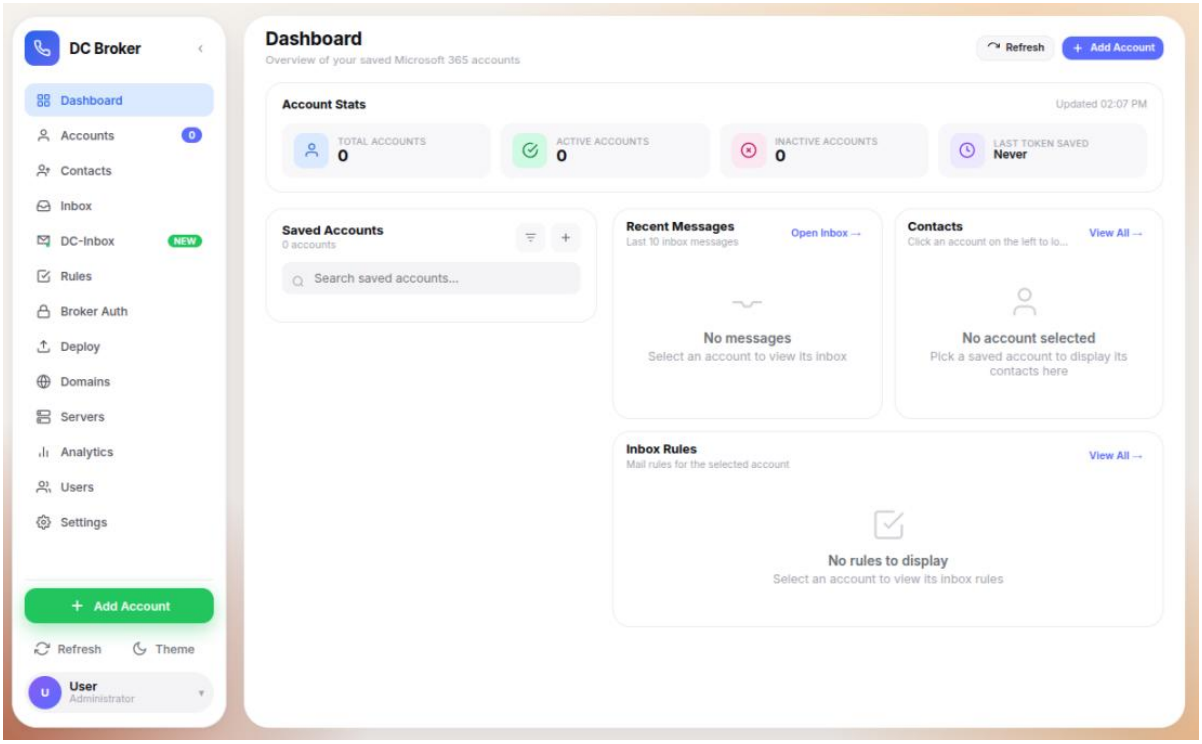


Figure 12 DC-Broker - general dashboard

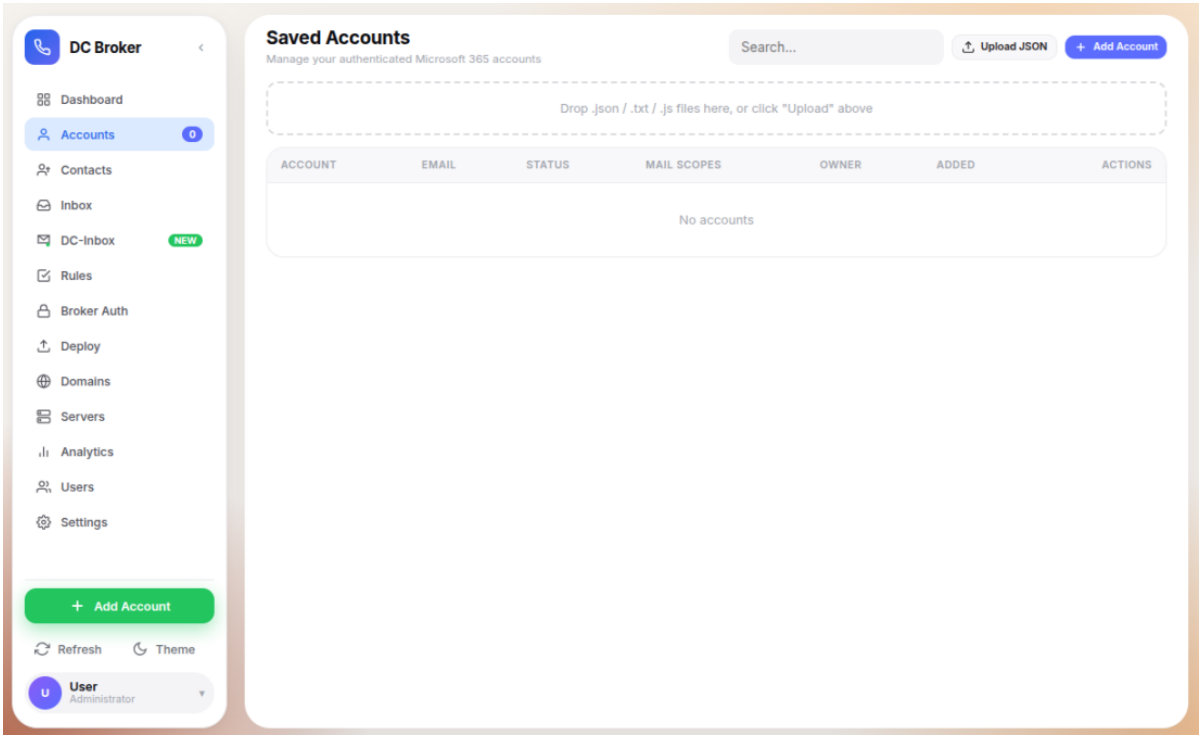


Figure 13 DC-Broker - saved account view

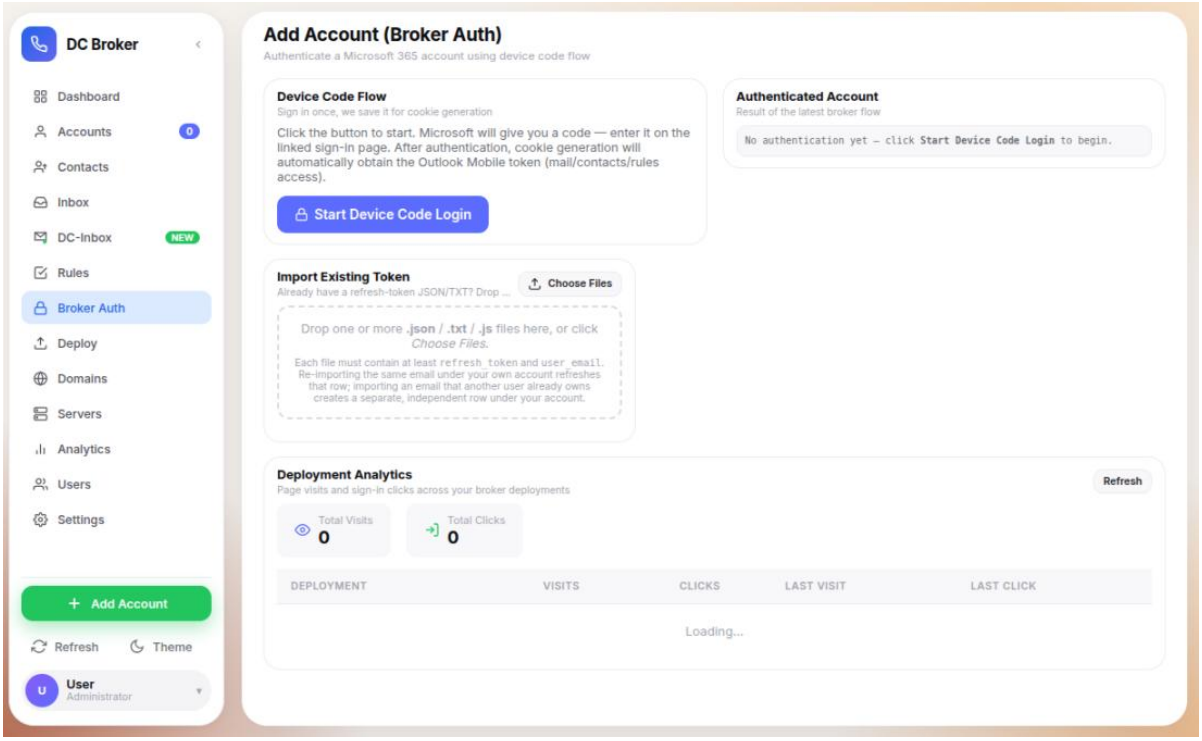


Figure 14 DC-Broker - adding accounts via Device Code Login

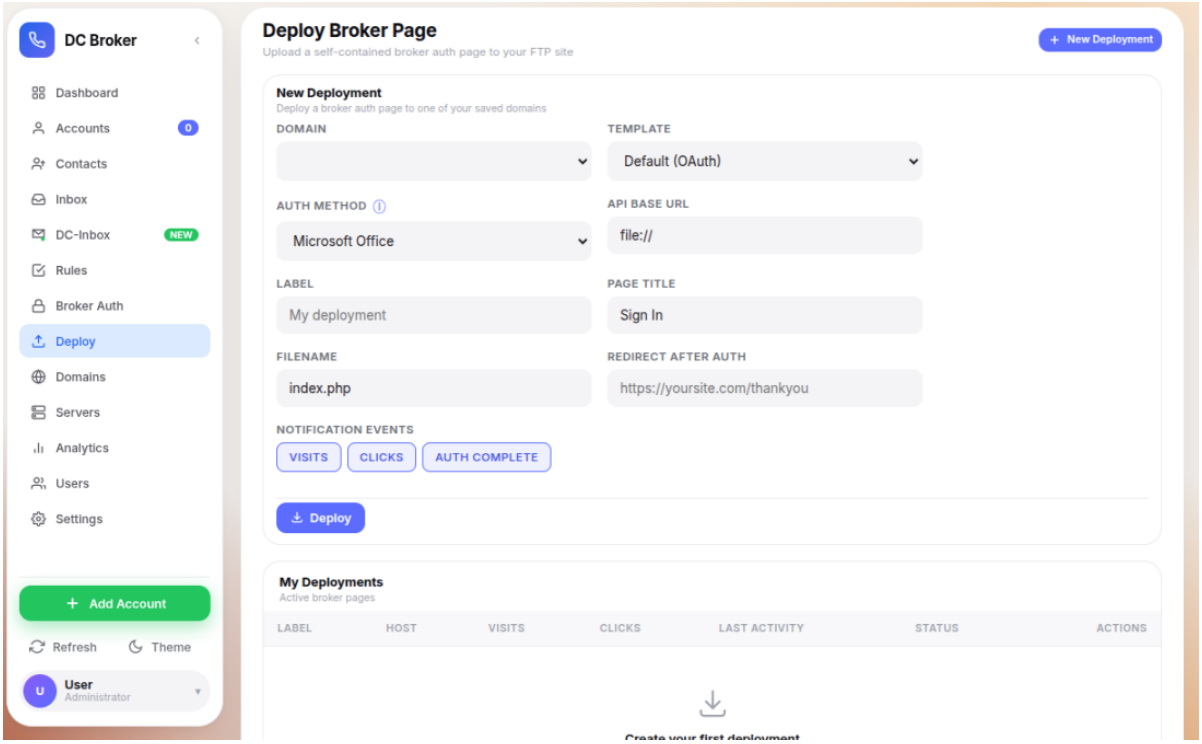


Figure 15 DC-Broker - new deployments page

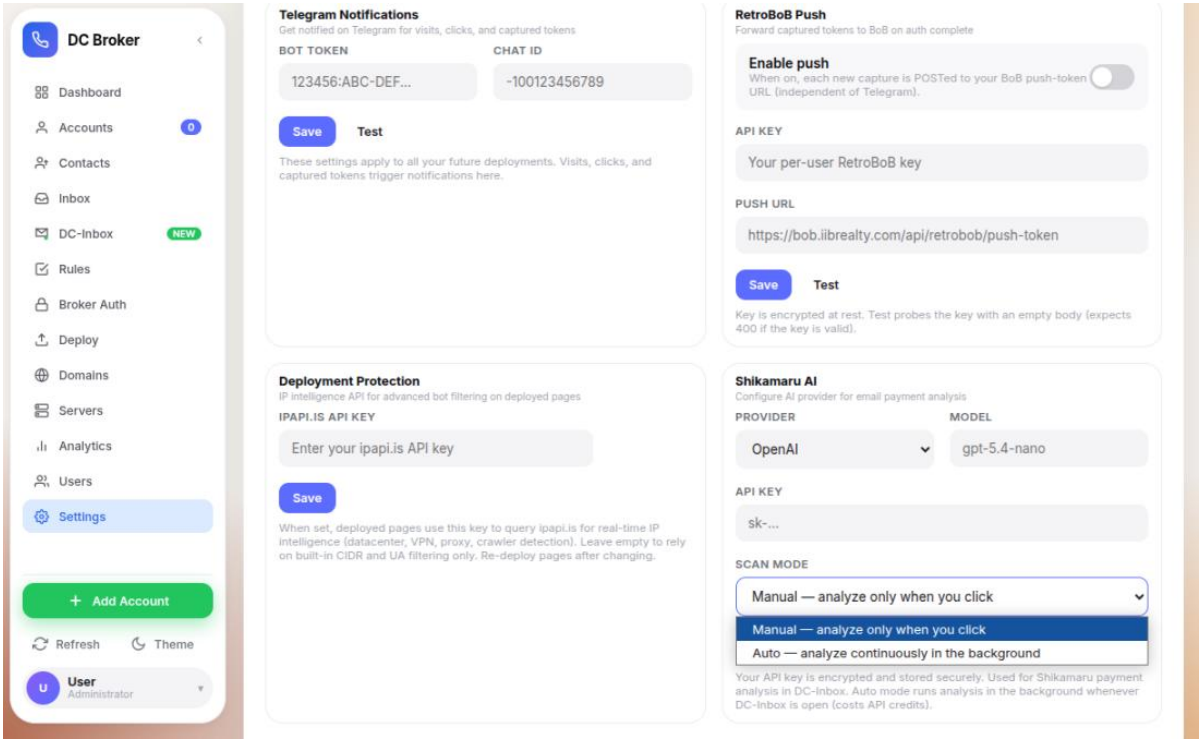


Figure 16 DC-Broker - settings include Telegram notifications and Shikamaru AI settings

4.2 Credential Stealing

4.2.1 Device Code Flow Phishing

DC Broker Dashboard automates this entire flow through its Broker Auth page:

- Operator clicks *"Start Device Code Login"* → `POST /api/broker/start`
- Backend initiates a device code request to Microsoft (possibly using a specific OAuth client ID / app registration to appear as a legitimate Microsoft application)
- A `user_code` is displayed in a styled "device code box" with the verification URL
- The operator delivers the code to the victim (via phishing email, social engineering, SMS, etc.)
- Victim visits `microsoft.com/devicelogin` and enters the code; completes MFA
- Dashboard polls `GET /api/broker/poll?flow_id=<id>` until status becomes `complete`
- `POST /api/broker/save` persists the refresh token to the backend database

The poll-and-display loop includes: - Real-time status feedback in the UI - Visual pipeline steps: *Device Code* → *MFA Wait* → *Token Received* → *Cookie* → *Mail Token* - Error categorization (MFA failures, tenant policy blocks, device join requirements)

4.2.2 Token Import

Beyond device code flow, the platform supports direct import of existing tokens via two mechanisms:

JSON/TXT refresh token import:

- Operator drops a `.json` or `.txt` file exported from any source
- File must contain at minimum `refresh_token` and `user_email`
- Backend validates and ingests the token
- Re-importing the same email refreshes the existing record; importing a different email creates a new record

Cookie JS import (`/api/import-cookie-js`):

- Operator drops a `.js` cookie file (as would be output by AiTM proxies or browser extension extractors)
- Backend auto-extracts the Outlook refresh token embedded in the cookie JS
- Inbox syncing begins automatically

This import capability means DC Broker functions as an aggregator, a central management console for tokens obtained by any means, including tools like Evilginx, Modlishka, browser stealers, infostealers, and manual extraction.

4.3 Token Management and Persistence

4.3.1 Token Lifecycle

Each acquired token is stored in the backend database and presented in the Accounts panel with the following tracked fields:

Field	Description
`user_email`	Victim's M365 email address
`user_name`	Display name
`token_id`	Internal unique identifier
`added_at`	Timestamp of acquisition
`proxy_country`	Assigned proxy geolocation
`owner_name`	Which operator owns this token
`Status`	`active` / `expired` / `mfa` (MFA-blocked) / `unknown`
`sync_error`	Last synchronization error if any
`mfa_blocked`	Boolean; MFA policy preventing re-use
`mfa_reason`	Specific reason code

4.3.2 Batch Status Monitoring

The dashboard continuously polls `GET /api/tokens/status` to maintain real-time visibility of the entire token fleet. Tokens that have expired or become MFA-blocked are prominently flagged, allowing operators to prioritize re-phishing or discard stale credentials.

4.3.3 EML Sync (Proactive Email Harvesting)

A dedicated background sync system (`/api/eml-sync/*`) proactively downloads and caches emails from all active accounts:

- Configurable sync window (number of days)
- Per-account sync status with error reporting
- Manual retry for failed accounts
- Cached EML files stored server-side for offline access
- Sync progress displayed as: *"Syncing X of Y accounts… (Z fully cached)"*

This proactive harvesting ensures that even if a token expires, operators retain access to the downloaded email archive.

4.4 Cookie Generation: PRT and nSSO Pipelines

4.4.1 Primary Refresh Token (PRT) Cookies

A PRT cookie is a special Microsoft session artifact that represents a device-bound authentication state. Generating a PRT cookie from a refresh token requires simulating a device registration interaction with Microsoft's identity platform. DC Broker automates this via a pipeline:

Trigger: Operator clicks *"Generate Cookie"* on an account
Flow: `POST /api/pipeline` → poll `GET /api/pipeline/<job_id>`
Output: JavaScript cookie string that can be injected into a browser to assume the victim's authenticated session

Pipeline steps tracked in the UI:

1. Token validation
2. Device join and/or registration
3. Primary Refresh Token acquisition
4. Cookie extraction
5. Mail token provisioning

The pipeline output is a JavaScript cookie injection snippet, ready to paste into DevTools Console or a browser extension.

4.4.2 Nonce-less SSO (nSSO) Cookies

DC Broker explicitly implements a second, more powerful cookie type:

> *"Generate a reusable nSSO (nonce-less SSO) cookie from the Outlook Mobile token. Unlike the regular cookie, this one can be used multiple times for the full PRT lifetime (~14–90 days)."*

Standard M365 session cookies are nonce-bound which means they are single-use and invalidate after one redemption. The nSSO cookie bypasses this mechanism, providing a persistent authenticated session valid for the full PRT lifetime of 14 to 90 days depending on tenant Conditional Access configuration.

This represents a significant tactical advantage: operators can access a victim account repeatedly over weeks without re-phishing or re-authenticating.

4.4.3 Outlook Mobile Token

A distinct "mail token" (Outlook Mobile OAuth token) is obtained as part of the pipeline. This token has scopes including functionality like `mail.read`, `mail.send`, `contacts.read`, `mailboxsettings.readwrite` (enabling inbox rule manipulation).

The Outlook Mobile token is documented in the platform as:

> *"After authentication, cookie generation will automatically obtain the Outlook Mobile token (mail/contacts/rules access)."*

4.5 Mailbox Access and Intelligence Gathering

4.5.1 Outlook-Style Inbox Reader

The platform provides a full three-pane Outlook-like email reader across all managed accounts simultaneously. This is architecturally significant: operators can read the emails of dozens or hundreds of victim accounts from a single interface.

Folder navigation: Inbox, Sent Items, Junk Email, Deleted Items, Drafts, Archives, Custom folders, monitored keyword folders (virtual, populated by server-side scan)

Message capabilities: Full message body rendering (HTML and plain text), Thread grouping (known as conversation view), Attachment viewing and download (PDF, images, Office documents, archives), and of course `.eml` download for offline forensic-style analysis. There's also option to search across all accounts simultaneously for specific keyword or phrase.

4.5.2 Live Fetch

A *"Fetch Live Now"* function (`POST /api/inbox/live-fetch`) triggers a proxied, real-time Graph API call against the victim's mailbox:

> *"The request goes through the per-token proxy and primes the EML download with up to 3 server-side retries."*

This bypasses the cached/synced view and retrieves the most current mailbox state, including emails received in the last few minutes — critical for time-sensitive BEC interceptions.

4.5.3 Keyword Monitoring and Alerting

The platform implements a keyword monitoring system across all victim mailboxes. Operators can define keywords like: *"wire transfer"*, *"invoice"*, *"password"*, *"urgent payment"*, *"verification code"*. The call of `POST /api/inbox/monitor/scan-now` triggers an immediate scan for those keywords. Matching emails are surfaced in a dedicated virtual folder, in which results are paginated and filterable.

This capability automates intelligence extraction across large account portfolios, enabling operators to rapidly identify high-value accounts (e.g., those receiving wire transfer instructions or password reset emails).

4.5.4 Contact Enumeration

Contact enumeration builds a graph of the victim's trusted relationships. This data directly enables internal spear-phishing (messages from a trusted colleague's compromised account), vendor impersonation BEC and lateral phishing to the victim's contact network.

The endpoint of `GET /api/contacts/unified` aggregates contacts from all managed accounts into a unified list with CSV export (`GET /api/contacts/unified/csv`). Per-account contact lists are also available.

4.5.5 Inbox Rule Manipulation

The token's `mailboxsettings.readwrite` scope grants write access to inbox rules. DC Broker displays existing inbox rules via `GET /api/tokens/<id>/rules`. While the current version of the frontend only reads rules, the underlying API access potentially enables:

- Creation of forwarding rules to exfiltrate all incoming email to attacker-controlled addresses
- Rules to auto-delete security alert emails (hiding the compromise from the victim)
- Rules to mark password reset emails as read (preventing victim notification)

This capability is foundational to persistent BEC operations where the victim must remain unaware of the compromise for an extended period.

4.5.6 Tagging System

Operators can apply custom tags to messages across accounts, enabling workflow organizations such as tagging accounts under active exploitation, marking messages containing payment details and coordinating between multiple operators on the same account.

4.6 Phishing Infrastructure Deployment

4.6.1 FTP-Based Deploy System

DC Broker includes a complete phishing page deployment subsystem. Operators can:

- Select a deploy template (broker authentication page design) from `GET /api/deploy-templates`
- Configure authentication method: Device Code Flow or Refresh Token (direct capture)
- Configure a redirect URL for post-authentication victim redirect
- Select a target FTP domain from the saved domain list
- Deploy with `POST /api/ftp/deploy`

The result is a public URL to the deployed phishing page — ready to send to victims.

4.6.2 Domain / FTP Management

Operators maintain a library of FTP-accessible domains:

```
POST /api/domains → { name, host, port, user, password, remote_path }
```

Each domain represents a server where broker pages can be deployed. FTP connection testing is available via `POST /api/ftp/test`. The interface supports multiple saved domains, enabling rapid rotation of phishing infrastructure.

4.6.3 Visitor Analytics

Deployed pages report back visitor telemetry:

Metric	Endpoint
Visit/click/auth events	`GET /api/deployments/analytics`
Summary statistics	`GET /api/deployments/analytics/summary`
Per-deployment stats	`GET /api/deployments/stats`

Operators can configure which events trigger notifications: - Visit — victim loaded the phishing page - Click — victim interacted - Auth Complete — victim successfully authenticated (token captured)

This analytics layer provides real-time campaign tracking, analogous to commercial phishing kit management platforms.

4.6.4 Notification on Capture

When a victim completes authentication on a deployed broker page, the platform can:
1. Send a Telegram notification with capture details

2. Automatically POST the token to a RetroBoB push endpoint

3. Log the capture in the deployments analytics database

The operator receives near-real-time notification of each successful credential capture, enabling rapid follow-up exploitation before the victim notices suspicious activity.

4.7 Token Forwarding and Integration with External Platforms

4.7.1 RetroBoB Integration

DC Broker integrates with an external platform referred to as RetroBoB:

> *"Forward captured tokens to BoB on auth complete." "When on, each new capture is POSTed to your BoB push-token URL."*

The default endpoint is:

`https://bob.iibrealty.com/api/retrobob/push-token`

Each operator configures their own `bob_api_key` and `bob_api_url`. This indicates RetroBoB is a separate, multi-tenant service to which DC Broker feeds tokens. The operational purpose is token aggregation: a single actor may use DC Broker for the acquisition phase and RetroBoB for the exploitation phase, allowing role separation within a criminal organisation (phishing operators vs. BEC operators).

4.7.2 Telegram Notification Pipeline

Real-time Telegram notifications are configured per-operator: - `telegram_bot_token` + `telegram_chat_id` - Events: visits, clicks, auth completions - Test notification function available

Telegram is the standard communication channel for cybercriminal operations due to its encryption, anonymity features, and API accessibility. The use of Telegram bots for real-time capture notification is consistent with commodity phishing kit design patterns observed since 2020.

4.7.3 Shikamaru AI Integration

A third-party AI system (referred to as Shikamaru, configured with an API key in `sk-xxx` format consistent with OpenAI-compatible APIs) is integrated for automated email intelligence analysis:

- `POST /api/shikamaru/analyze` — bulk-analyzes emails across the managed account fleet
- `GET /api/shikamaru/items` — returns AI-generated intelligence items with urgency classification (Critical / High / Medium / Low)
- `GET /api/shikamaru/summary` — aggregated intelligence summary
- `GET /api/shikamaru/timeline` — chronological intelligence timeline
- `GET /api/shikamaru/vendors` — vendor/contact relationship analysis
- Per-item Deep Research function: `GET /api/shikamaru/items/<id>/research`

This is an operationally significant capability: AI-powered automated analysis of victim mailboxes at scale enables operators to process large account portfolios and surface high-value intelligence (payment flows, vendor relationships, internal approval chains) without manual review of each inbox. The platform effectively functions as an AI-augmented intelligence collection system targeting victim organizations.

4.8 Proxy Infrastructure

4.8.1 Per-Token Proxy Assignment

Each victim token is associated with a proxy country:

- Operators (or admins) are assigned a `proxy_country` that is applied to all their tokens
- `GET /api/proxy-countries` returns the list of available proxy locations
- A global *"Random Proxy"* mode (`/api/app-settings`) randomly selects from the full proxy list rather than matching by country

Purpose: Microsoft's Conditional Access policies can block sign-in attempts from countries or IP ranges inconsistent with the victim's usual location. By routing token usage through a proxy matching the victim's known geography, operators evade location-based anomaly detection.

4.8.2 Proxy Application to Live Fetch

The live fetch feature (`POST /api/inbox/live-fetch`) explicitly routes the Graph API call through the per-token proxy:

> *"Proxied Graph call → caches metadata → kicks off EML download with up to 3 retries server-side."*

This ensures that real-time mailbox access is indistinguishable from legitimate access originating from the victim’s expected geography.

5 DC-Inbox – a dedicated mailbox intelligence and exploitation interface

DC-Inbox is a purpose-built, Outlook-style webmail reader designed exclusively to monitor, search, and exploit Microsoft 365 mailboxes whose OAuth refresh tokens were stolen via the companion DC Broker platform. Unlike DC Broker, which handles credential acquisition and infrastructure deployment, DC-Inbox focuses entirely on the post-compromise intelligence phase: real-time email reading across dozens or hundreds of victim accounts simultaneously, AI-powered analysis of financial and operational communications, email composition and impersonation from victim accounts, and a sophisticated conversation-tracking and bookmarking system. DC-Inbox shares the same Flask backend as DC Broker and is co-deployed on the same server, representing a mature operational separation of concerns within a criminal toolchain.

DC-Inbox is the dedicated intelligence-exploitation companion to DC Broker. While DC Broker covers the full attack lifecycle (phishing, token storage, cookie generation, deployment), DC-Inbox is a specialist tool focused exclusively on what you do after you have the tokens.

The platform is described in its own source code as:

"Original code authored for the dc-broker app. Calls existing /api/ endpoints — no backend changes required."*

This confirms that DC-Inbox is a second frontend layered on top of the same DC Broker backend, providing a dedicated, optimized mailbox interface. Operators who need to manage phishing campaigns use DC Broker; operators who need to process the harvested inbox data use DC-Inbox. This role separation is a hallmark of mature, structured criminal operations.

The platform's title is simply "DC-Inbox", presented as a clean, Outlook-faithful design with the Microsoft Fluent/Segoe design language — intentionally mimicking a legitimate enterprise mail client to reduce cognitive friction for operators browsing victim mail.

Similar tech stack as DC Broker Dashboard. Uses same api endpoints (single shared backend). It's co-located alongside the DC Broker Dashboard inside /dc-inbox path.

5.1 Application overview

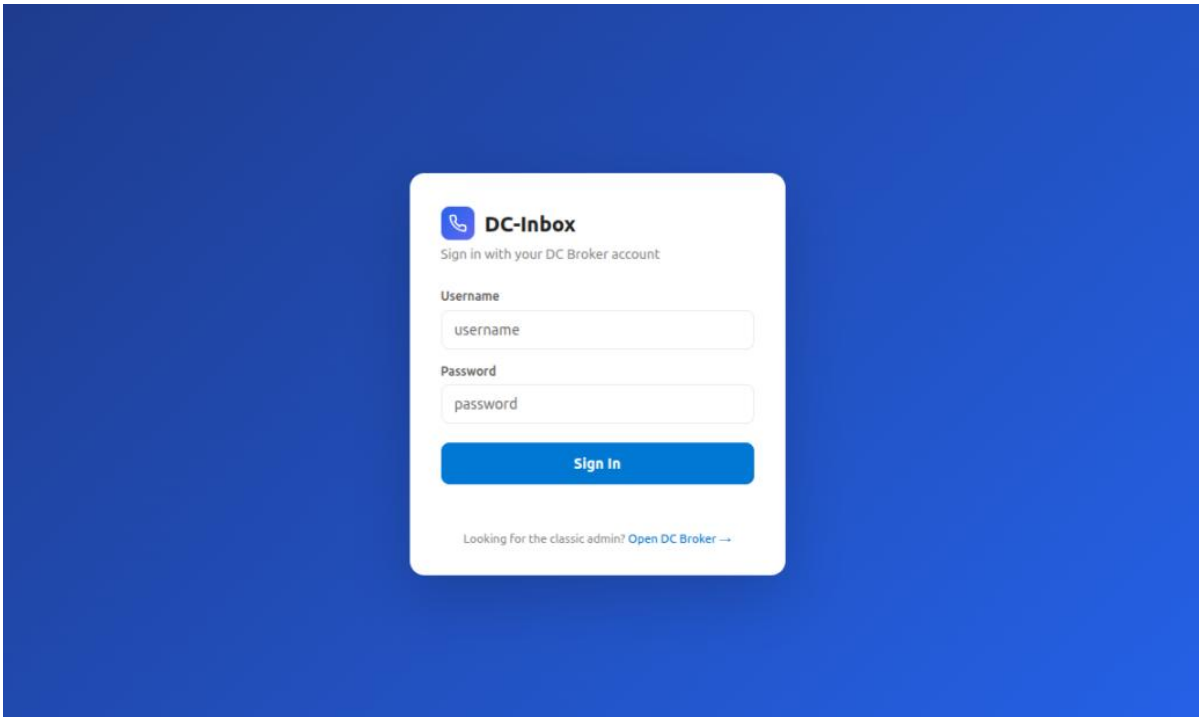


Figure 17 DC-Inbox login panel with an optional redirection to DC Broker

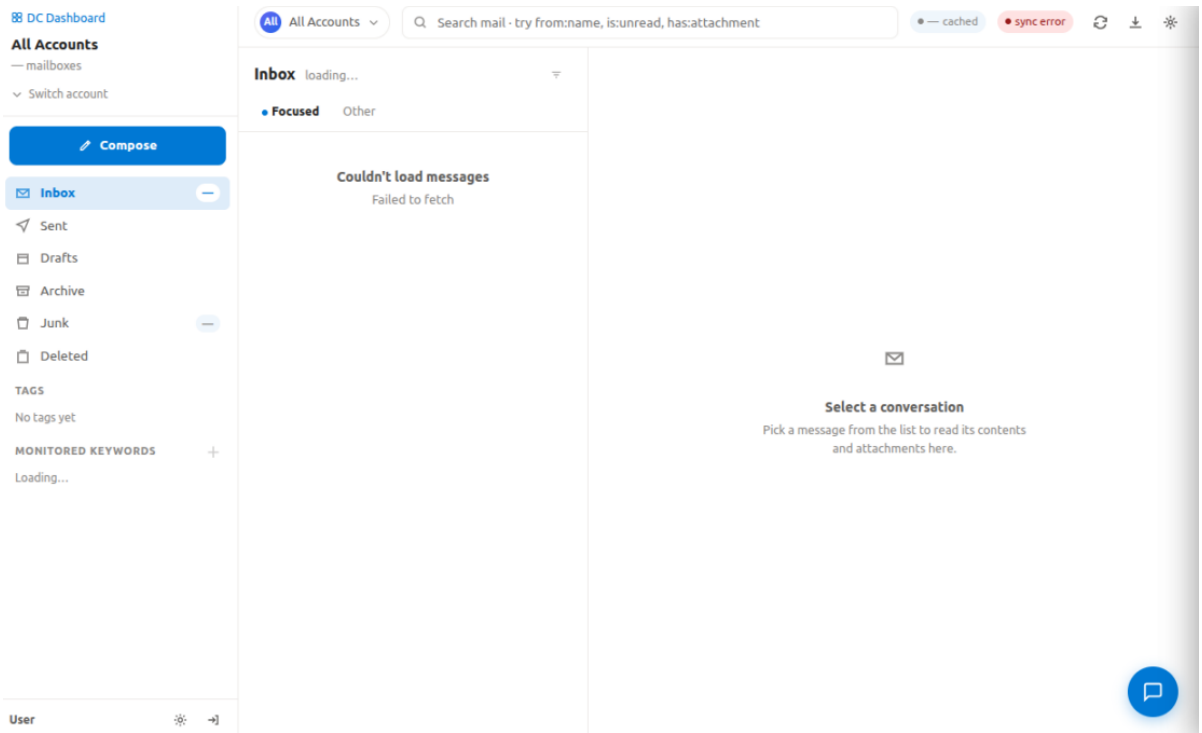


Figure 18 DC-Inbox - inbox view

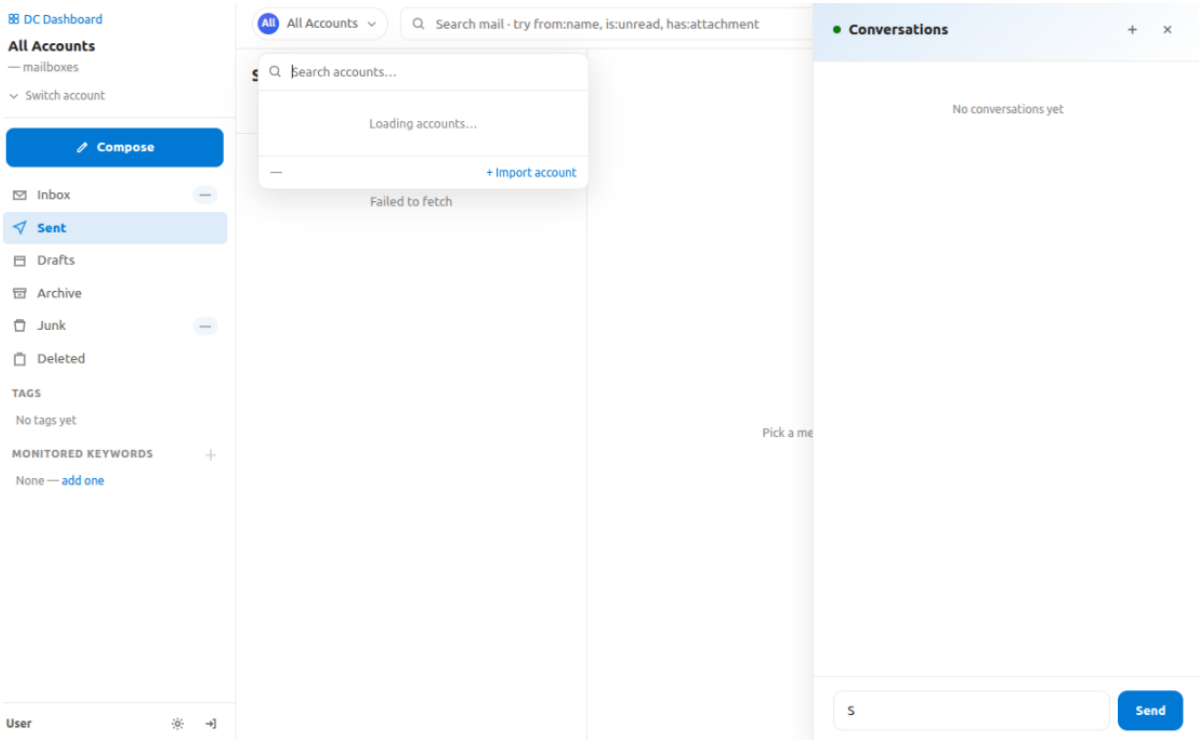


Figure 19 DC-Inbox Conversations window

5.2 Account Management and Token Ingestion

5.2.1 Account Switcher

The top-left account picker (ox-acct-pop) lists all managed victim accounts. Each account shows: - Email address and display name - Status dot: active (green), mfa (amber), expired (red), pending (grey) - Unread message count badge

Operators can switch between victim accounts with a single click or view all accounts simultaneously ("All inboxes" mode) in a merged chronological feed.

5.2.2 Token Import Drawer

A dedicated import drawer (DC.openImport()) accepts two token types:

JSON token files: - Drag-and-drop one or more .json files from the broker export pipeline - Each file requires at minimum refresh_token and user_email - Multiple files processed in a single drop

JS cookie files: - Drop .js cookie files (AiTM proxy output format) - Auto-extraction of the embedded Outlook refresh token - Inbox syncing begins immediately after import - Progress shown: *"Importing X / Y accounts…"*

The platform enforces an import_cap limit (GET /api/app-settings → import_cap) — a configurable maximum number of accounts the operator is permitted to import, suggesting this is a licensed or quota-controlled service.

5.3 Mailbox Reading: The Three-Pane Reader

DC-Inbox provides a full three-pane Outlook-style email interface:

Folder list	**Message list**	**Message reader**
+ Keywords	(threaded rows)	+ Attachments
+ Stars	+ Pagination	+ Toolbar
+ Tags	+ Search	Right rail: Shikamaru AI

Folder Navigation

Standard M365 folders are presented: - **Inbox** — with unread count - **Junk Email** — with unread count
- **Sent Items**, **Deleted Items**, **Drafts**, **Archives** - **Custom folders** — user-defined server-side folders - **Monitored keyword folders** — virtual folders populated by server-side scan - **Starred conversations** — bookmarked high-value threads

Message List

Messages are displayed in threaded (conversation) view with: - Sender name and email - Subject line - Body preview snippet - Timestamp (relative and absolute) - Account indicator (in "all inboxes" mode) - Star button for bookmarking - ESP (Email Service Provider) detection badge

Pagination loads additional messages on scroll (loadMoreList()), supporting large mailboxes.

Message Reader

Clicking a message: 1. Marks it as read locally (POST /api/inbox/mark-read-local) 2. Fetches the full message from the backend (GET /api/tokens/<tid>/messages/<mid>) 3. Renders HTML body (or plain text in `<pre>`) 4. Shows toolbar: Reply, Reply All, Forward, Tag, Download EML, Delete, Move

HTML body rendering: The email HTML body is rendered directly in the reader pane without sanitisation — an XSS risk identical to that documented for DC Broker's inbox (see analyzed.md).

5.4 Live Fetch: Real-Time Proxied Graph Access

The source comment describes the "Fetch Live Now" operation as

"The request goes through the per-token proxy and primes the EML download with up to 3 server-side retries."

This is significant: - The request is proxied through the victim's geolocation proxy, bypassing Microsoft location-based anomaly detection - Up to 3 automatic retries are attempted server-side, making the fetch robust against transient network failures - Results are cached locally for offline browsing - The UI blocks with a loading state during the fetch and shows the live result immediately on completion

This feature enables operators to retrieve emails received seconds ago — critical when monitoring a victim account during an active business email compromise (e.g., waiting for a payment confirmation to arrive).

5.5 Conversation Threading and Bookmarking

5.5.1 Thread View

DC-Inbox groups messages by conversationId into threaded conversation cards, mirroring Outlook's conversation view. The thread reader (renderThreadReader()) stacks individual message cards vertically, each expandable with a click. All messages in a thread share a single toolbar.

5.5.2 Star System (Conversation Watchlist)

Operators can "star" any conversation — bookmarking it as high-value:

POST /api/inbox/stars { conversation_id, token_id, subject, sender, account_email, message_id }
GET /api/inbox/stars → list all starred conversations
GET /api/inbox/stars/<star_id>/hits?limit=100 → fetch all messages in starred thread
DELETE /api/inbox/stars/<star_id> → unstar

Before starring, the platform calls GET /api/inbox/resolve-conversation to resolve the conversationId and pull all known messages in that thread.

Operational use: An operator monitoring 200 victim accounts spots an email thread about an upcoming wire transfer. They star the conversation. The starred view then tracks every new message in that thread across all future syncs, enabling the operator to follow the conversation as it progresses without manually searching for it — automating surveillance of high-value financial threads.

5.5.3 Tags

Operators can apply custom text labels to messages:

GET /api/inbox/tags → list all tags (filtered by account)
POST /api/inbox/tags/<tag>/messages → apply tag
DELETE /api/inbox/tags/bulk-delete → remove tag(s)

Tags appear as virtual folders in the left panel, allowing organization of messages by campaign, victim, or exploitation status.

5.6 Email Composition and Impersonation

DC-Inbox provides a full Quill.js rich-text compose drawer - a WYSIWYG email editor for sending email directly from victim accounts.

5.6.1 Compose Modes

Mode	Description
new	New message from selected victim account
reply	Reply to current message (quoted original included)
replyAll	Reply-all to current thread
forward	Forward message to new recipients

The compose drawer mirrors Thunderbird-style quoting behavior:

"Original message is included below your reply — edit it like in Thunderbird."

5.6.2 Send Mechanism

POST /api/tokens/<token_id>/send (multipart/form-data)

The request uses FormData (multipart) to support file attachments. Fields include: - to, cc, bcc — recipient chip arrays - subject — composed subject line - body — Quill HTML output - reply_to_id — original message ID (for reply threading) - Attached files

The sender identity is the victim's Microsoft 365 account — the email appears to originate from victim@company.com in all recipient mail clients. This is the core BEC email fraud capability.

5.6.3 BEC Scenarios Enabled

- Payment diversion: Reply to an active invoice thread, changing payment details
- Internal impersonation: New message from CFO's account to AP team authorizing urgent transfer
- Vendor impersonation: Forward a legitimate vendor email with modified bank details
- Credential harvest pivot: Send phishing emails from the compromised account to the victim's contact list (trusted source, high click rate)
- Thread hijacking: Reply into ongoing M&A or contract negotiations from a participant's account

5.7 Attachment Handling and Document Preview

DC-Inbox implements a multi-format attachment preview modal:

GET /api/tokens/<tid>/messages/<mid>/attachments → list attachments
GET /api/tokens/<tid>/messages/<mid>/attachments/<aid> → download/preview

Supported preview types:

Type	Rendering method
PDF	`<iframe>`
Images (PNG/JPG/GIF/WebP)	`<img>` tag
Plain text / code	`<pre>`
Office documents (DOCX/XLSX/PPTX)	Parsed server-side → rendered HTML
Archives (ZIP/RAR/7z)	Listing
Email files (EML/MSG)	Parsed and rendered
Other	Fallback download link

Operational use**:** Operators can read invoices, contracts, financial statements, and other sensitive attachments directly in the browser without downloading — reducing forensic footprint on the operator's machine.

5.8 Keyword Monitoring and Alerting

The keyword monitoring system (/api/inbox/monitor/*) enables automated surveillance across all victim mailboxes:

GET /api/inbox/monitor/keywords → list keywords
POST /api/inbox/monitor/keywords → add keyword
DELETE /api/inbox/monitor/keywords/<id> → remove keyword
GET /api/inbox/monitor/keywords/<id>/hits?token_id=... → fetch matching messages
POST /api/inbox/monitor/scan-now → trigger immediate scan

Each keyword appears as a virtual folder in the left panel. Clicking a keyword folder loads all emails matching that keyword across all monitored accounts. Operators paginate through results with a "Load more" button.

Typical operator keywords: wire transfer, payment, invoice, bank details, password, reset, verification, MFA, code, urgent, CEO, CFO, executive, acquisition, contract, NDA, confidential, salary, bonus, payroll, BitLocker, recovery, admin.

This system enables efficient triage of large account portfolios: rather than reading every email, operators define interest patterns and the platform surfaces matching messages automatically.

5.9 Shikamaru: AI-Powered Mailbox Intelligence

The Shikamaru right-rail panel (accessed via the shikamaru tab in the right sidebar) provides AI-automated intelligence extraction across the entire managed account fleet.

5.9.1 Analysis Workflow

POST /api/shikamaru/analyze { "scan_days": 30 } → trigger bulk AI analysis
GET /api/shikamaru/progress → poll analysis progress
GET /api/shikamaru/pending-work?scan_days=30 → preview work queue

The analysis scans configurable number of days of email across all active accounts, passing email content to an AI backend (Shikamaru service, using an sk-... format OpenAI-compatible API key).

5.9.2 Intelligence Items

Each item returned has: Urgency tier: Critical / High / Medium / Low, Subject and sender, Account email (which victim account the email belongs to**), Conversation ID** (clickable to open the full thread), Status: pending / resolved / dismissed

GET /api/shikamaru/items?limit=N&offset=N&status=pending → paginated items
GET /api/shikamaru/summary → urgency counts

View	Endpoint	Description
List	/api/shikamaru/items	All items, filterable by urgency
Timeline	/api/shikamaru/timeline	Chronological event timeline
Vendors	/api/shikamaru/vendors	Vendor/company relationship graph
Archive	Resolved + dismissed items	Historical intelligence record

5.9.3 Deep Research

Each intelligence item has a **"Research"** button:

POST /api/shikamaru/items/<id>/research (SSE stream)

This triggers a Server-Sent Events stream returning a deep-dive AI analysis of the specific email thread — identifying parties, amounts, timelines, action requirements, and potential fraud vectors.

5.9.4 Star Integration

Shikamaru items are directly actionable: each item has a **star button** that triggers starConversation(), immediately adding the related email thread to the operator's watch list.

Operational significance: Shikamaru effectively functions as an AI analyst working 24/7 on behalf of the threat actor, scanning hundreds of victim mailboxes for high-value

intelligence and surfacing actionable items prioritised by urgency. An operator managing 500 victim accounts can review the morning's Critical and High items in minutes rather than hours.

5.10 AI Conversational Chat Interface

Beyond automated analysis, DC-Inbox includes a conversational AI chat interface:

```
GET    /api/inbox/ai/chats                    → list conversations
POST   /api/inbox/ai/chats                    → create new conversation
GET    /api/inbox/ai/chats/<id>               → load conversation history
POST   /api/inbox/ai/chats/<id>/message       → send message
DELETE /api/inbox/ai/chats/<id>               → delete conversation
```

The chat drawer is accessed via a floating action button (FAB). Message format:

```
{
  "message": "What is the current status of the Q3 payment approval for Acme Corp?",
  "context": { "token_id": 42, "message_id": "AAMk..." }
}
```

The AI response may include sources — references to the specific emails used to form the answer. If a message is currently open in the reader, its token_id and message_id are automatically included as context.

Conversations are persisted server-side (multi-turn, with history). Chat conversations are listed in a side panel with titles derived from the first message and can be deleted individually.

Operational use: An operator can ask questions like: - *"Summarise all pending wire transfer approvals across all accounts"* - *"Which accounts have emails from bank security teams in the last week?"* - *"What are the payment terms in the contract thread with [vendor]?"* - *"Identify all executives who have emailed from their personal accounts"*

This provides a natural-language interface to the stolen email corpus, dramatically accelerating the intelligence-to-action cycle.

5.11 EML Synchronisation and Offline Cache

```
GET  /api/eml-sync/status                  → per-account sync status + error count
POST /api/eml-sync/<token_id>/retry        → retry failed sync
```

The EML sync subsystem runs continuously in the background, downloading raw .eml files for all accounts. Status tracked per account: - synced_emls — count of successfully cached messages - error_msg — last error (shown as a sync error row with retry button)

Purpose of offline cache: 1. Messages remain accessible after a token expires 2. Faster rendering (no live Graph API call required) 3. Full message body available for AI analysis 4. .eml download for forensic-quality offline copy

5.12 Contact Management

DC-Inbox exposes contacts per account (read-only in the inbox view, consistent with DC Broker's contact enumeration):

GET /api/tokens/<tid>/contacts → per-account contact list

Contacts are displayed as a scrollable list with name, email, and metadata. The platform sorts and paginates contact results.

5.13 Inbox Rule Management

GET /api/tokens/<tid>/rules → list inbox rules

Inbox rules are displayed in the reader panel for the selected account. The interface is read-only in DC-Inbox (rule creation is handled in DC Broker), but provides visibility into: - Rules the operator has already created (forwarding, auto-delete, auto-categorise) - Rules the victim has created (which may reveal security concerns or filter patterns)

6 BoB V2 – An adversarial mass email platform

BoB V2 ("Broker of Browsers", subtitled "Tier-2 Outlook") is a self-hosted, multi-user web panel built for the systematic exploitation of stolen Microsoft 365 OAuth session cookies. Operating as a downstream consumer of credentials produced by phishing frameworks such as DC Broker, BoB V2 automates contact harvesting from victim mailboxes, manages an account fleet with per-account proxy routing, deploys HTML email campaigns through compromised accounts, and provides full inbox rule management to sustain persistent, covert access. This paper provides a full technical dissection of the platform.

The Two-Phase Compromise Model

Modern Microsoft 365 account takeover operations are typically split into two phases:

- Phase 1 (Acquisition): Steal OAuth tokens or session cookies via Device Code phishing, AiTM proxy frameworks (Evilginx, Modlishka), or browser infostealers. Platforms like DC Broker serve this role.
- Phase 2 (Exploitation): Monetise the stolen access — read emails, harvest contacts, and send phishing or BEC emails from the compromised account. BoB V2 is purpose-built for this exploitation phase.

BoB V2 is explicitly described as "Tier-2 Outlook", clearly positioning it as a downstream platform that consumes credentials from a "Tier-1" acquisition layer. Its architecture and user interface are optimised for bulk email operations: loading cookie files, extracting contacts at scale, and firing personalised HTML email campaigns from dozens of compromised accounts simultaneously.

Relationship with DC Broker

BoB V2 is tightly integrated with DC Broker via the **RetroBoB push-token** mechanism: when DC Broker captures a new token, it can automatically POST it to BoB V2's POST /api/retrobob/push-token endpoint, making the account immediately available in BoB without any manual operator action. This creates a fully automated pipeline from phishing to exploitation.

Similarly to previously described apps, the technology behind is Flask and HTML+JS.

6.1 Application Overview

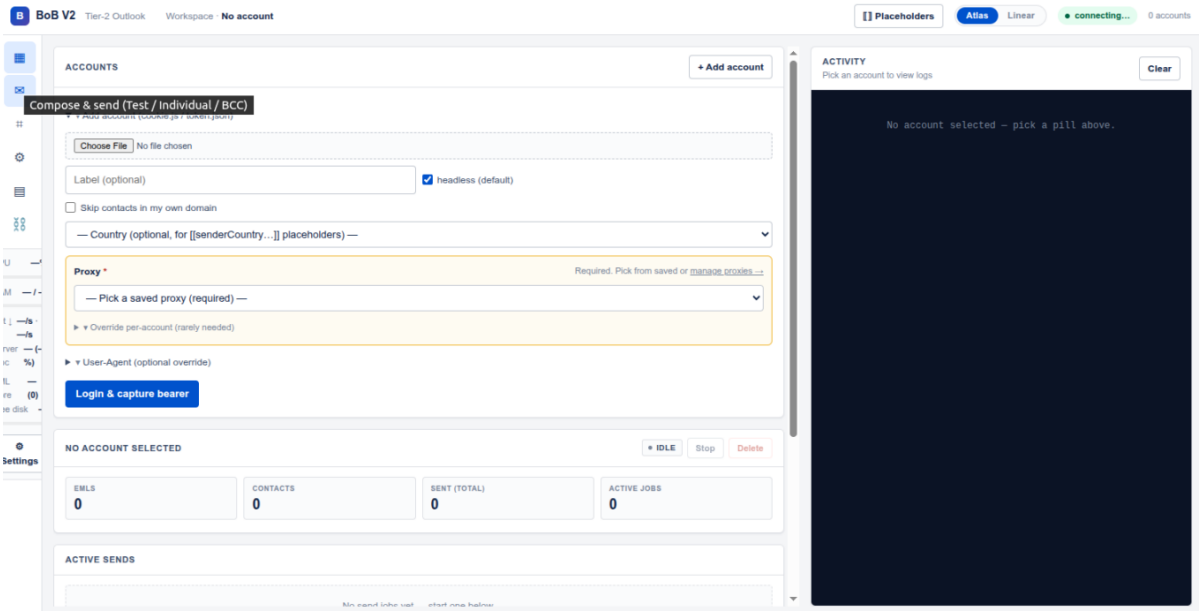


Figure 20 BoB V2 Tier-2 Outlook view. It get's a bit broken when log-in prompt is skipped.

6.2 Account Management and Cookie Ingestion

6.2.1 Cookie File Import

Accounts are added to BoB V2 by dropping a cookie file:

- Accepted formats: .js, .json, .txt
- The file contains an Outlook/M365 session cookie or refresh token exported from a browser or extraction tool
- Backend validates and ingests the session

The import UI comments reveal operational details:
"*↻ Re-auth button. The JS sets dataset.targetId — account to re-auth in the change handler.*"
This supports a re-authentication workflow: when a cookie expires, the operator drops a new cookie file targeting the existing account record, refreshing the session without losing harvested contacts or configuration.

6.2.2 Account Status Lifecycle

Status	Visual Indicator	Meaning
Active/Ready	Green ring	Cookie valid; ready for operations
Inactive/Error	Solid red ring	Cookie expired or invalid; needs re-auth
Running	Animated	Active extraction in progress

6.2.3 Per-Account Settings

Each account stores: - Label (operator-defined name) - Country (for template placeholder resolution: [[senderCountry]], [[senderCountryCode]], [[senderCountryISO]]) - Proxy assignment (required for every account) - skip_own_domain — exclude contacts from the same domain as the compromised account (anti-detection measure) - Active template ID - Redirect URL (for phishing link personalisation) - gate_secret_key (64-char hex for bot-gate protected phishing pages) - Auto-rule configuration

6.2.4 Headless Browser Automation

Accounts use Chromium in headless mode for cookie-based authentication:
"Run Chromium without a visible window. Default ON — uncheck only if a cookie isn't logging in cleanly and you need to watch the browser interact with Outlook."
The platform launches a headless Chromium browser authenticated with the victim's cookie, then drives Microsoft Graph API calls through it. This makes the access pattern appear as a real browser session rather than a programmatic API client, evading some anomaly detection systems.

6.3 Proxy Infrastructure

Every account in BoB V2 requires a proxy. This is enforced at the UI level:
"Proxy is required for every account."
The mandatory proxy model ensures that all mailbox access is routed through a geographically appropriate IP address, defeating location-based Conditional Access policies and reducing the risk of triggering Microsoft's anomaly detection.

6.3.1 Saved Proxy Library

Operators maintain a library of saved proxies configured with: - Host, port, username, password, scheme (HTTP/HTTPS/SOCKS5) - Country assignment - POST /api/proxy-test — verify connectivity before assignment
Proxies are stored in user settings (GET/PUT /api/auth/settings) and available in a dropdown when adding or reconfiguring accounts.

6.3.2 BitLaunch VPS Integration

BoB V2 includes **direct integration with BitLaunch** — a VPS provider that accepts cryptocurrency payments and is commonly used by threat actors:

- POST /api/bitlaunch/validate-key / save-key
- GET /api/bitlaunch/balance
- GET /api/bitlaunch/providers / options/{host_id}
- POST /api/bitlaunch/create-proxies — automatically provision VPS instances as proxies
- GET/DELETE /api/bitlaunch/servers

This integration allows operators to spin up new proxy servers directly from within BoB V2, funded by cryptocurrency, without leaving the panel interface. The significance is substantial: the platform provides a complete, integrated infrastructure management system for maintaining rotating, anonymous proxy infrastructure.

6.4 Contact Harvesting

6.4.1 Extraction Modes

Contact extraction is the primary intelligence-gathering operation in BoB V2. The platform supports multiple extraction modes:

- **Quick API extract** — Uses Microsoft Graph API to retrieve contacts quickly
- **EML extract** — Downloads raw email files from the mailbox, parses sender/recipient metadata to build a contact graph
- **Combined** — Runs both methods and merges results

The extraction logs are accessible at GET /api/account/{id}/logs.

6.4.2 ESP Classification

A key feature is **ESP (Email Service Provider) classification** of harvested contacts. The platform automatically classifies each harvested email address by its email provider (Gmail, Outlook, Yahoo, corporate, etc.).
This classification is used during the send phase to **filter recipients by ESP**, allowing operators to: - Send only to Gmail users (to test deliverability on Google's infrastructure) - Exclude corporate email addresses (reducing detection risk from security teams) - Target specific email providers where phishing lure is most effective

6.4.3 *skip_own_domain* Anti-Detection Filter

When enabled, contacts whose email domain matches the compromised account's own domain are automatically discarded during extraction. Example: an account alice@acme.com will not accumulate bob@acme.com in its contacts. This reduces the risk of sending phishing emails to colleagues of the victim — who might recognize unusual activity and report it.

6.4.4 Third-Party List Import

Beyond extracted contacts, operators can import third-party contact lists: - POST /api/contacts/third-party/import - POST /api/contacts/third-party/preview - GET /api/contacts/third-party/{list_id}
This allows bulk-purchased email lists to be combined with organically harvested contacts for larger-scale campaigns.

- 6.5 Contact Management
- PATCH /api/account/{id}/contacts/{email} — edit individual contact metadata
- POST /api/account/{id}/contacts/delete — bulk delete by email list
- POST /api/contacts/delete — cross-account bulk delete
- GET /api/contacts/export — export contacts (scope, account filter, include_generic flag)

6.5 Email Composition and Sending

6.5.1 Three Sending Modes

BoB V2 offers three distinct email sending workflows, each serving a different operational purpose:

- *Test Send*

POST /api/account/{id}/send-test
Sends a single test email from the specified account to a controlled address. Used to verify: - Template rendering (placeholder resolution) - Deliverability (does the email land in inbox or spam?) - Cookie validity (does the account still have an active session?)

- *Individual Send*

POST /api/account/{id}/send-individual
Sends one email to a specific recipient from the specified account. Used for targeted BEC operations where a single, carefully crafted email is sent to a high-value target.

- *BCC Blast*

POST /api/account/{id}/send-bcc
Sends a bulk email with all recipients in BCC from a single compromised account. This is the primary mass-phishing capability: one email, many recipients, all appearing to receive an individual message from the compromised sender’s legitimate email address. The BCC method is particularly effective for phishing because: - Recipients see the email from a real, known organization’s email domain - SPF/DKIM authentication passes because the email genuinely originates from the legitimate mail server - No phishing infrastructure is required — the email is sent via Microsoft’s own servers

6.5.2 ESP Filtering at Send Time

When composing a BCC blast or individual campaign, operators can filter the recipient list by ESP:

- *“Send only to ESP: [filter selection]”*
- *“BCC only to ESP: [filter selection]”*

Buttons allow selecting all providers or none, then manually toggling individual providers. This granular filtering optimises campaign targeting and delivery rates.

6.5.3 Recipient Override

A "Recipient override" feature allows operators to temporarily substitute a real recipient list with a test address, enabling full campaign rehearsal without actually sending to victims.

6.6 Template Engine

6.6.1 HTML Email Templates

BoB V2 maintains a template library (GET/POST/PUT/DELETE /api/templates). Each template: - Contains HTML email body - Supports a rich placeholder system - Can be set as the "active template" per account - Can be previewed with placeholders resolved using the active account's data

6.6.2 Placeholder System

The template engine supports dynamic placeholders that are resolved at send time using victim account data:

Placeholder	Value
[[senderCountry]]	Full country name of the compromised account
[[senderCountryCode]]	Phone country code (e.g. +47)
[[senderCountryISO]]	ISO country code (e.g. NO)
[[senderCompany]]	Company name extracted from account
[[base64Email]]	Base64-encoded victim email (for tracking links)
[[sender]]	Sender name/email

Placeholders also work inside redirect URLs, allowing per-recipient personalised links:
https://phishing-site.com/?u=[[base64Email]]
This creates unique, trackable phishing URLs for each recipient — a standard technique for link-click tracking in phishing campaigns.

6.6.3 Template Preview and Rendering

- GET /api/templates/{id}/preview — rendered HTML preview
- GET /api/templates/{id}/pdf_source + POST /api/templates/{id}/pdf_preview — PDF rendering (for attachment-based campaigns)
- GET /api/templates/{id}/ics — calendar invite generation (ICS attachment phishing)
- GET /api/templates/{id}/asset/{name} — embedded asset access

The ICS and PDF capabilities indicate the platform supports multiple email-borne attack vectors beyond simple HTML links — calendar invite abuse and malicious document delivery.

6.7 Inbox Rule Automation

6.7.1 Auto-Rule (Automatic on Account Add)

A defining operational feature: BoB V2 can automatically push an inbox rule into a victim's mailbox the moment the cookie is ingested:
"The moment BoB captures the bearer token from a fresh login (or resume), this rule is pushed straight into that mailbox."
The auto-rule is configured per-user in settings and applied to every new account automatically, enabling instant persistent access without a separate manual step.

6.7.2 Rule Configuration Interface

The settings panel provides a full inbox rule composer with:
Conditions (When the message matches…): Subject contains, From address, Has attachments, Body contains, Recipient domain
Actions (…then do this): Automatic forward to attacker-controlled address, routes all matching emails to the attacker in real time, Move to folder, Copy to folder, Mark as read, prevents victim from seeing notifications, Delete, destroys evidence, Apply categories - Mark as important (to prioritize certain messages).

6.7.3 Persistent Rule Management

Beyond auto-rules, a full rules management interface allows: - GET /api/account/{id}/rules — list all rules - POST /api/account/{id}/rules — create rule - PATCH /api/account/{id}/rules/{rule_id} — modify rule - DELETE /api/account/{id}/rules/{rule_id} — delete rule - GET /api/account/{id}/folders — list available folders for rule targeting
Automatic forwarding rules are the cornerstone of long-term BEC operations: even if the original cookie expires, incoming emails continue to flow to the attacker's address.

6.8 RetroBoB Integration (Token Push Receiver)

BoB V2 exposes a push-token API endpoint that receives tokens pushed from upstream acquisition platforms (DC Broker, custom scripts, or other phishing frameworks):
POST /api/retrobob/push-token
Configuration is stored in user settings under the retrobob key. This creates a fully automated, zero-touch pipeline:
[DC Broker captures token]
→ POST /api/retrobob/push-token
→ BoB V2 adds account automatically
→ Auto-rule pushed to victim mailbox

→ Contact extraction begins
→ Account ready for send campaigns

The operator receives value without any manual intervention between capture and exploitation.

7 BoB V3 - A Fully Modular, AI-Augmented Microsoft 365 Exploitation Platform

BoB V3 is the third-generation iteration of the Broker of Browsers platform and represents a significant architectural and capability maturation over BoB V2. Moving from a single-file monolith to a modular, multi-file architecture, V3 introduces a fully integrated RetroBoB automation pipeline with AI (LLM) company inference, a multi-account PRO Mode campaign engine with ESP filtering and Bot Gate protection, BitLaunch cryptocurrency-funded VPS proxy provisioning, a multi-tenant API key management system for the RetroBoB service, granular audit logging with admin impersonation controls, and a send-protection anti-phishing detection layer. This paper provides a complete technical dissection of BoB V3's architecture, capabilities, and threat model.

This confirms bob.iibrealty[.]com is the canonical hosting infrastructure for this platform, and V3 is an explicitly versioned release path. The domain is the same endpoint DC Broker and BoB V2 reference as the RetroBoB push-token destination, establishing it as a central hub in this criminal ecosystem.

7.1 Application Overview

This section will contain screenshots displaying particular dashboard from inside, showcasing different features available to the attacker.

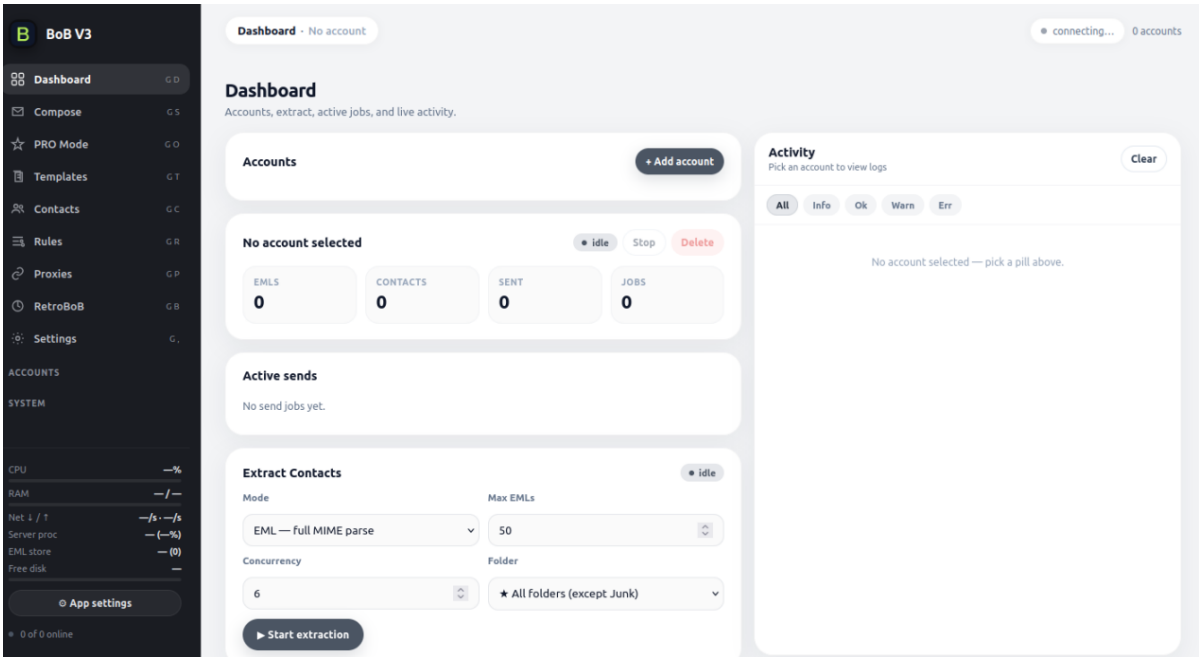


Figure 21 BoB V3 - Dashboard view

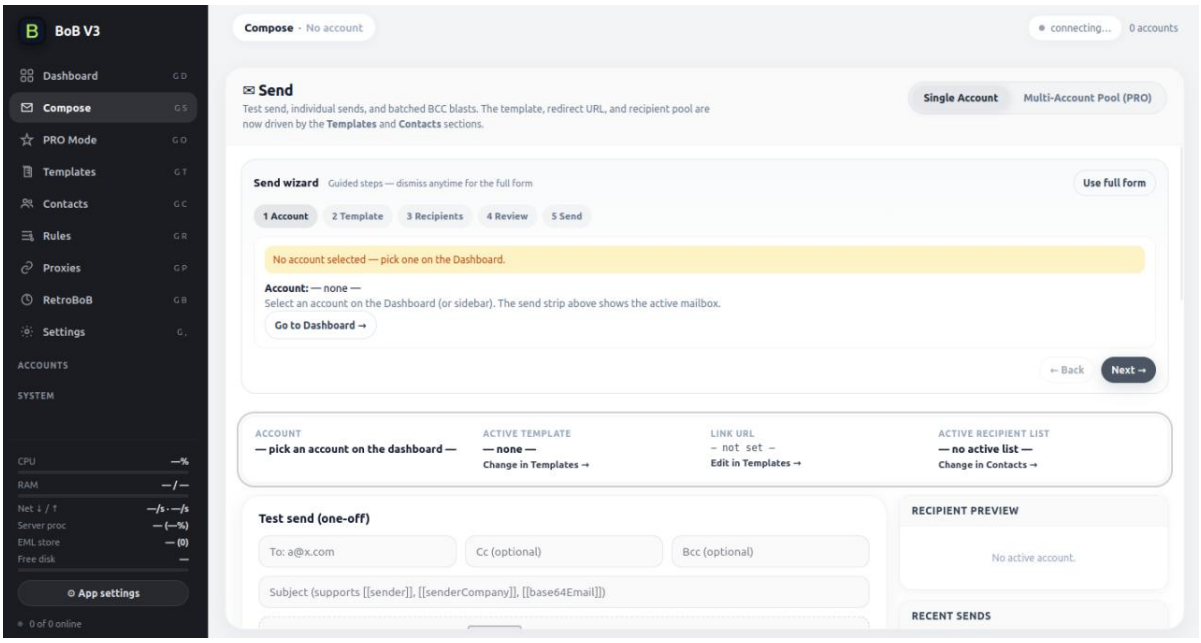


Figure 22 BoB V3 - Email Compose section

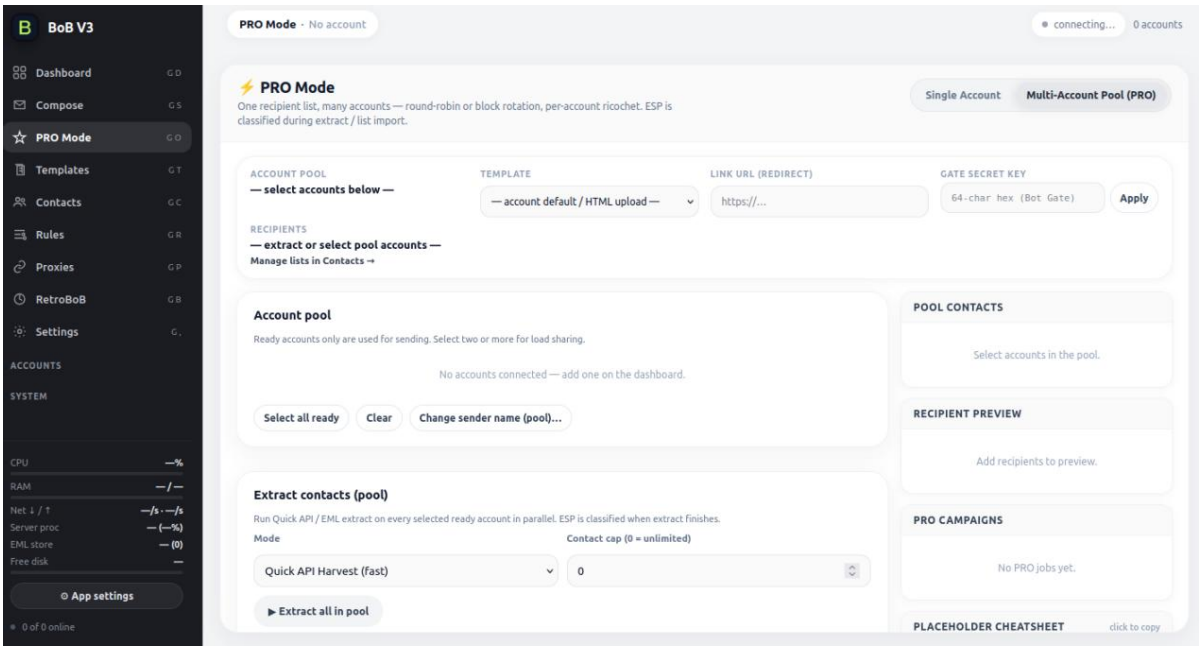


Figure 23 BoB V3 - the PRO mode

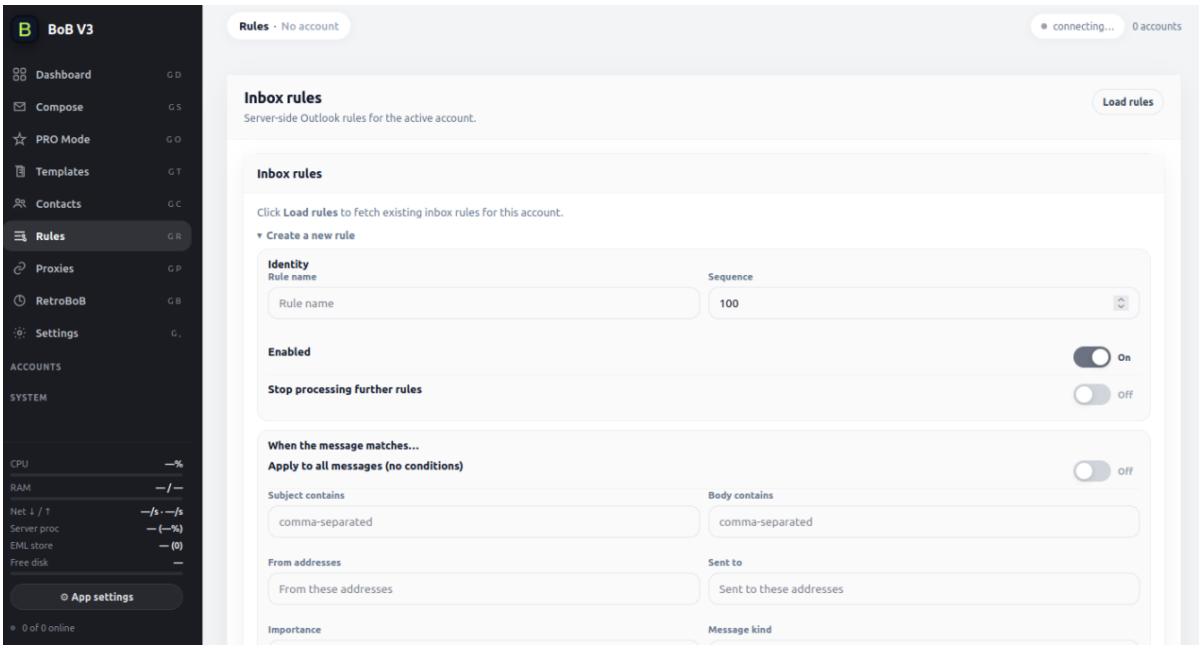


Figure 24 BoB V3 - inbox rules load and view panel

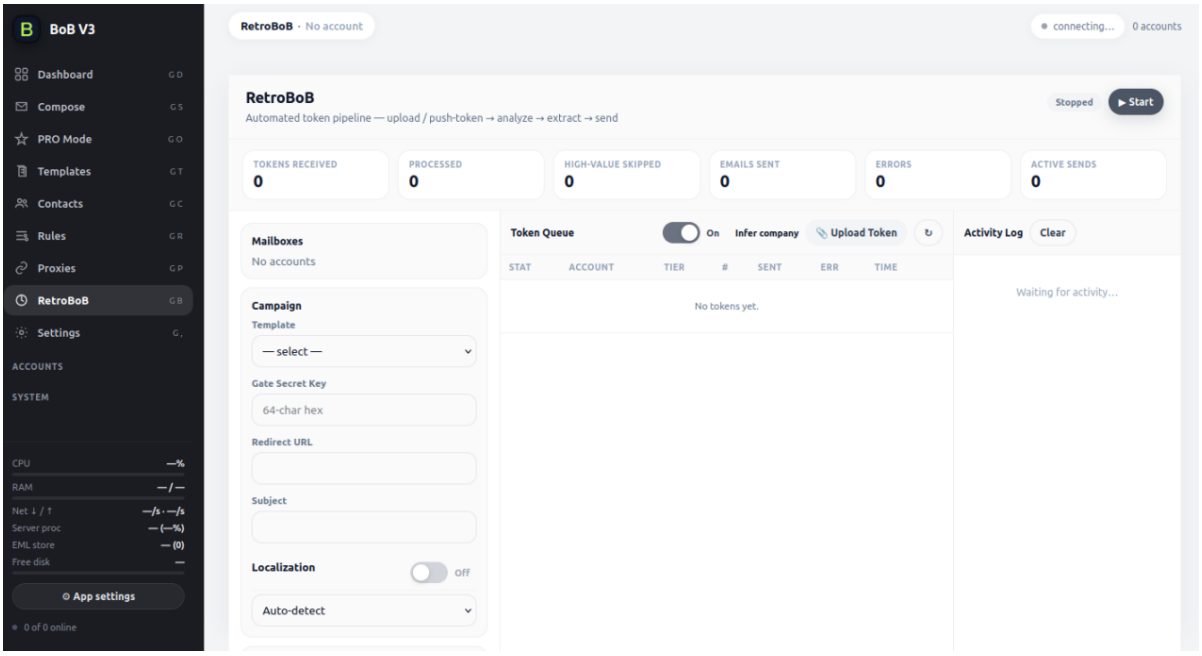


Figure 25 BoB V3 - RetroBoB connection

7.2 Architectal Evolution from V2 to V3

From Monolith to Modular - BoB V2 was delivered as a single 6,091-line HTML file. BoB V3 transitions to a split architecture:

Component	V2	V3
Shell/HTML	bob_v2.html (all-in-one)	BoB V3.html (2,433 lines — shell + mount points)
Auth/Admin	Inline	auth_admin.js
Dashboard	Inline	dashboard.js
Compose/Send	Inline	compose.js
Contacts	Inline	contacts.js
Rules	Inline	rules.js
Templates	Inline	templates.js
Proxies	Inline	proxies.js
Settings	Inline	settings.js
RetroBoB	Basic push-receiver	retrobob.js (full pipeline module)
Styles	Inline	app.css

Each JS module explicitly documents its API surface at the top of the file — a professional software engineering practice that doubles as internal documentation for the criminal tooling team.

7.3 Proxy Infrastructure and BitLaunch Integration

V3 promotes the proxy module to a full first-class page in the navigation. The BitLaunch integration is identical to V2 but now documented explicitly in `proxies.js`:

```
POST /api/bitlaunch/validate-key
POST /api/bitlaunch/save-key
GET  /api/bitlaunch/key-status
DELETE /api/bitlaunch/key
GET  /api/bitlaunch/balance
GET  /api/bitlaunch/options/{host_id}
GET  /api/bitlaunch/providers
POST /api/bitlaunch/create-proxies    Body: {count, regionID, sizeID, imageID, ho
stID}
GET  /api/bitlaunch/servers
DELETE /api/bitlaunch/servers/{bl_server_id}
POST /api/account/{id}/assign-saved-proxy
```

The `POST /api/bitlaunch/create-proxies` endpoint is the most significant: by specifying `count`, `regionID`, `sizeID`, and `imageID`, an operator can provision a batch of geographically distributed VPS instances as proxies in a single API call, paid for with the BitLaunch API key (which is typically funded by cryptocurrency). The provisioned servers then appear in the proxy library and can be bulk-assigned to accounts.

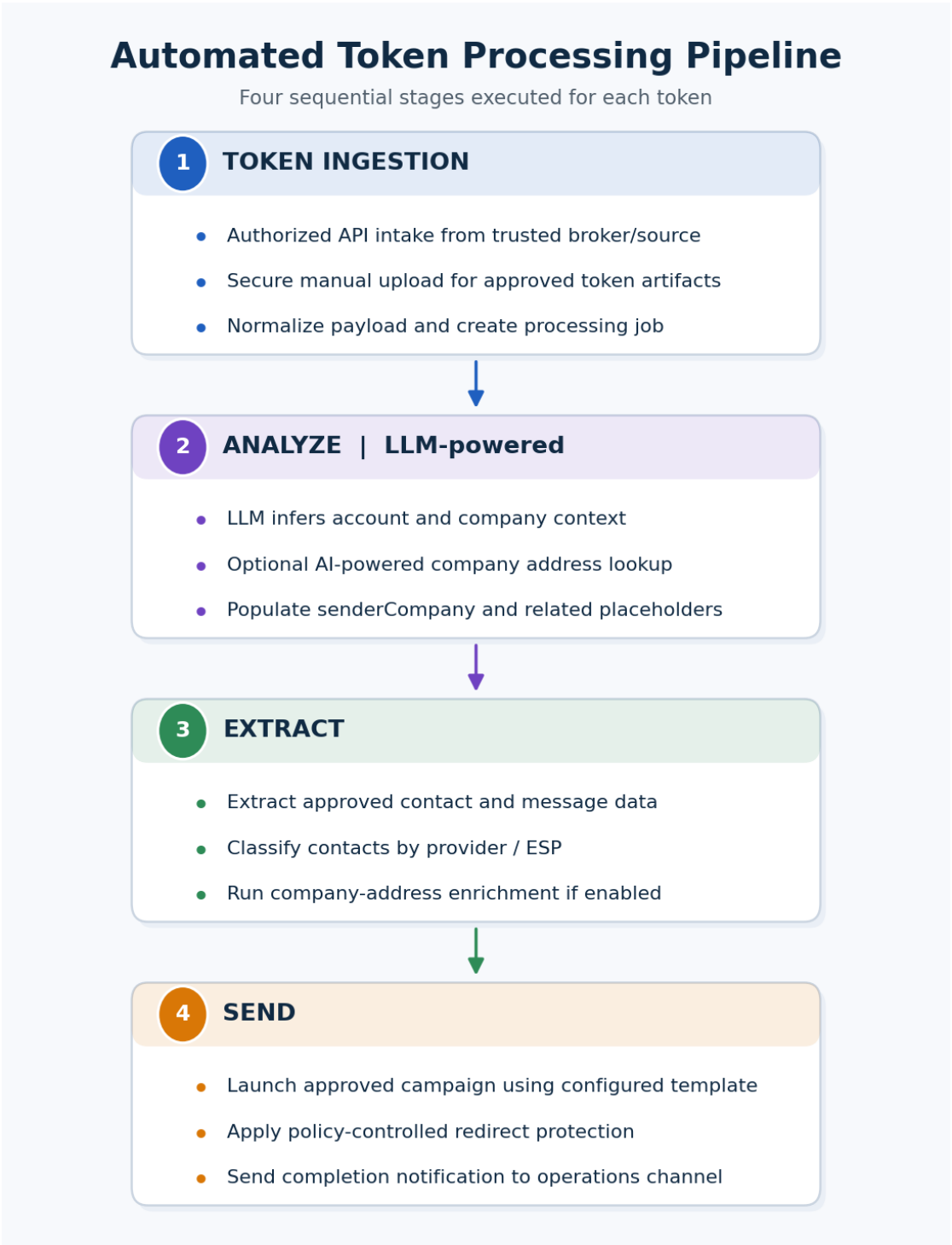
Operational implication: An operator running a campaign targeting a UK-based organisation can provision UK-geolocated proxies via BitLaunch, assign them to the stolen UK accounts, and run all mailbox access through locally-appearing IP addresses, all from within the BoB V3 panel.

7.4 RetroBoB Automation Pipeline (V3 Native)

The RetroBoB module in V3 is vastly more capable than the basic push-token receiver in V2. It is now a complete, managed automation pipeline with its own dedicated page in the navigation.

7.4.1 Pipeline Description

"Automated token pipeline — upload / push-token → analyze → extract → send". The pipeline has four automated stages that execute sequentially for each token:


Automated Token Processing Pipeline
Four sequential stages executed for each token
1
TOKEN INGESTION
Authorized API intake from trusted broker/source
Secure manual upload for approved token artifacts
Normalize payload and create processing job
2
ANALYZE | LLM-powered
LLM infers account and company context
Optional AI-powered company address lookup
Populate senderCompany and related placeholders
3
EXTRACT
Extract approved contact and message data
Classify contacts by provider / ESP
Run company-address enrichment if enabled
4
SEND
Launch approved campaign using configured template
Apply policy-controlled redirect protection
Send completion notification to operations channel

7.4.2 AI-Powered Company Inference

A critical V3 capability: **LLM-based automatic company inference**:

- Toggle: *"Infer company (AI)"*
- Toggle: *"Infer company address"*
- Configuration: LLM API key (`settingsRetroLlmKey`) stored in RetroBoB settings
- Test endpoint: `POST /api/retrobob/test-llm`

When enabled, the RetroBoB pipeline uses an LLM (OpenAI-compatible API, key in `sk-...` format) to automatically determine the victim's company name and address from the email account metadata. This populates `[[senderCompany]]`, `[[senderAddress]]`, and similar placeholders in the email template **without any manual operator input**.

Operational implication: A token pushed from DC Broker automatically triggers a complete pipeline that ends with a personalised phishing email sent from the victim's account to their contacts — with the company name and address correctly populated in the template — with zero operator interaction. The platform achieves nearly complete automation of the phishing-to-BEC pipeline.

7.4.3 Queue Management with Manual Override

The RetroBoB queue provides full operator control:

Queue Status	Description
`queued`	Waiting to begin processing
`analyzing`	LLM analysis in progress
`extracting`	Contact extraction in progress
`sending`	Campaign being sent
`paused`	Manually paused
`done`	Pipeline complete
`error`	Failed at a stage

Per-token actions: - ▶ Approve/Send — manually trigger send after review - Skip — skip this token - Pause/Resume — pause the pipeline at current stage - Cancel — abort processing

Global controls: `start`, `stop`, `pause`, `resume`, `cancel-all`

The operator can configure whether tokens auto-proceed through all stages or require manual approval at the send stage — enabling a review-before-send workflow for high-value targets.

7.4.4 Multi-Tenant API Key Management

V3 introduces a full API key management system for the RetroBoB service itself:

```
GET    /api/retrobob/api-keys        — list all issued keys
POST   /api/retrobob/api-keys        — issue new key
DELETE /api/retrobob/api-keys/{id}   — revoke key
```

This confirms that RetroBoB is operated as a **multi-tenant service** — different operators (customers) each have their own API key to push tokens to the pipeline. The BoB V3 panel is the management interface for both using and administering the service.

7.4.5 Telegram Notification Integration

`POST /api/retrobob/test-telegram` confirms Telegram notification is wired into RetroBoB. Operators receive real-time alerts for each pipeline stage completion and campaign send confirmation.

7.5 PRO Mode: Multi-Account Campaign Engine

PRO Mode is BoB V3's flagship new sending capability, described as:

> *"One recipient list, many accounts — round-robin or block rotation, per-account ricochet. ESP is classified during extract / list import."*

7.5.1 Account Pool

Operators select multiple ready accounts as a sending pool. PRO Mode distributes the recipient list across all pool accounts using:

- Round-robin rotation - each subsequent email sent from the next account in the pool
- Block rotation - a batch of emails sent from one account before rotating

7.5.2 Per-Account Ricochet

Each account in the pool has a ricochet configuration — a per-account redirect URL that the phishing link routes through before reaching the final destination. This allows: - Different tracking parameters per account - Different landing page variants A/B tested per account pool member - Per-account gate secret keys for bot filtering

7.5.3 Pool-Wide Controls

- `POST /api/pro/extract` — run extraction on all pool accounts simultaneously
- `POST /api/pro/campaign` — fire the campaign across the pool
- `POST /api/pro/test-send` — test from the pool
- `POST /api/pro/change-sender-name` — change sender display name across all pool accounts at once

- `POST /api/pro/apply-gate-key` — apply a single Gate secret key to all pool accounts
- `GET /api/pro/pool-contacts` — aggregate contacts from all pool accounts

7.5.4 Operational Impact

PRO Mode fundamentally changes the scale and evasion characteristics of email campaigns: - Distributing 10,000 emails across 20 accounts means 500 emails per account — below most per-account anomaly detection thresholds - Each email originates from a different `From:` address/domain — defeating blacklists targeting a single sender - Round-robin rotation mimics normal email traffic patterns

7.6 Bot Gate Protection System

V3 introduces a Bot Gate mechanism integrated into the phishing page redirect:

```
Gate secret key: [64-char hex string]
```

Applied via: - Per-account `gate_secret_key` in settings - `POST /api/pro/apply-gate-key` for pool-wide application

Purpose: A Bot Gate is a server-side filter on the phishing landing page that validates incoming visitors. When a victim clicks the phishing link, the redirect URL includes a cryptographic token derived from the gate key. The gate validates this token before serving the phishing page content. This prevents:

- Automated security scanners (URL reputation services, email security gateways) from fetching and categorising the phishing page
- Click-rate inflation by bots
- Premature phishing page takedown by security vendors who crawl URLs from email headers

The 64-character hex key provides sufficient entropy to prevent brute-force guessing. By applying the same gate key across all pool accounts, the gate can validate any link from the pool without per-account configuration overhead.

7.7 ESP Classification and Send Targeting

V3 retains and extends V2’s ESP classification system:

- ESP is classified during extraction and during third-party list import
- `POST /api/account/{id}/contacts/reclassify-esp` — re-run classification on existing contacts

- `POST /api/contacts/third-party/{list_id}/reclassify-esp` — re-classify a third-party list
- `POST /api/contacts/ensure-esp` — ensure all contacts have ESP labels
- `GET /api/esp/labels` — retrieve list of ESP label categories

At send time, operators can filter by ESP both for individual sends and BCC blasts, with the same per-provider toggle interface as V2.

7.8 Template Engine and ICS/PDF Capabilities

Templates in V3 are managed by a dedicated `templates.js` module with full CRUD:

```
GET/POST/PUT/DELETE /api/templates
GET    /api/templates/{id}
GET    /api/templates/{id}/preview
GET    /api/templates/{id}/pdf_source
POST   /api/templates/{id}/pdf_preview
GET    /api/templates/{id}/ics
GET    /api/templates/{id}/asset/{name}
POST   /api/templates/import/{template_id}
PUT    /api/account/{id}/settings     (active_template_id, redirect_url, gate_sec
ret_key)
```

The ICS (iCalendar) capability deserves special attention. Calendar invite abuse is a well-documented attack vector:

- A `.ics` file attached to an email opens as a calendar invite in Outlook
- The invite body can contain HTML with phishing links
- Recipients expect calendar invites from known contacts and are less suspicious than of email links
- Outlook renders calendar invite HTML in a less restricted context than email HTML

The **PDF** capability similarly supports document-based phishing — generating a PDF attachment that appears to be a legitimate business document with a phishing link embedded.

7.9 Contact Management and Third-Party Lists

V3's contact module is more feature-rich than V2:

Feature	Endpoint
Export contacts	`GET /api/contacts/export?scope=&account_id=&include_generic=`
Import third-party list	`POST /api/contacts/third-party/import`
Preview third-party list	`POST /api/contacts/third-party/preview`

Delete individual contacts	`POST /api/account/{id}/contacts/delete` body: `{emails}`
Cross-account bulk delete	`POST /api/contacts/delete` body: `{emails}`
Wipe all contacts	`DELETE /api/contacts`
Edit contact metadata	`PATCH /api/account/{id}/contacts/{email}`

The `include_generic` parameter on export controls whether generic addresses (`info@`, `support@`, `admin@`) are included — allowing clean deduplication for targeted campaigns versus mass campaigns.

7.10 Inbox Rule Automation

V3's auto-rule system gains additional rule condition toggles compared to V2:

New in V3: - *"Stop processing further rules"* — ensures the auto-rule runs before any existing rules, preventing victim-side rules from counteracting the attacker's rule - *"Apply to all messages"* — blanket capture of all incoming mail - *"Sent only to me"* — target only directly addressed messages (avoids CC/mailing list noise) - *"I'm in To"* / *"I'm in Cc"* — granular recipient position targeting - *"Permanent delete"* — non-recoverable deletion (bypasses Deleted Items, goes straight to recoverable items with short retention)

The *"Permanent delete"* action is the most aggressive new capability: it can destroy incoming security alert emails without leaving a trace in Deleted Items, making the compromise harder for the victim to discover during an incident investigation.

7.11 Send-Protection (AFP) Layer

V3 references AFP (Anti-Phishing Protection) settings stored in `auth/settings`:

> *"Telegram + LLM + push-token API keys. Campaign fields live on the RetroBoB page. […] send-protection"*

The `GET/PUT /api/auth/settings` endpoint manages `send-protection` configuration alongside auto-rule, proxy library, and RetroBoB settings. While the specific send-protection mechanism is not fully described in the client code, the context indicates it is a **pre-send validation layer** that checks whether the outgoing email campaign is likely to trigger anti-phishing detections on the recipient side (spam filters, link reputation checks). This is an operational evasion capability.

7.12 API Key Management for RetroBoB Service

V3 introduces multi-tenant API key issuance for the RetroBoB service, elevating it from an internal integration to an externally accessible service API:

```
GET    /api/retrobob/api-keys
POST   /api/retrobob/api-keys
DELETE /api/retrobob/api-keys/{key_id}
```

This means: - Other tools (DC Broker, custom scripts, other phishing frameworks) can push tokens to BoB V3's RetroBoB using an API key - Different customers/operators receive different API keys - API keys can be revoked without affecting other tenants - The BoB V3 admin can see all issued keys and their usage

This architecture is consistent with a commercial crimeware-as-a-service offering where BoB V3 is the backend service and multiple customers push tokens to it via the RetroBoB push-token API.

8 Conclusions

8.1 DC Broker

DC Broker represents a mature, operationally sophisticated platform purpose-built for the industrialized acquisition and exploitation of Microsoft 365 credentials. It occupies a unique position in the criminal tooling ecosystem by combining:

- Automated credential acquisition via Device Code flow phishing — no proxy infrastructure required, bypasses MFA
- Token management at scale — multi-operator, multi-tenant, with proactive EML harvesting
- Session hijacking via PRT and reusable nSSO cookie generation
- Full mailbox intelligence — real-time reading, contact enumeration, keyword alerting, AI analysis
- Email fraud execution — sending emails from victim accounts, reply injection
- Phishing infrastructure lifecycle management — deploy, track, and analyze campaigns
- Criminal ecosystem integration — Telegram, RetroBoB, AI APIs

The platform's design philosophy — professional UX, modular architecture, multi-operator support, and tight integration with the broader criminal tooling ecosystem — suggests a well-resourced threat actor with significant Microsoft 365 expertise and software engineering capability.

Defenders should treat Device Code flow as a primary attack vector requiring explicit Conditional Access controls, ensure Continuous Access Evaluation is enabled across all M365 tenants, and monitor for the behavioral and infrastructure indicators documented in this paper.

8.2 DC-Inbox (part of DC Broker Dashboard)

DC-Inbox represents a mature, specialized post-compromise platform that transforms stolen Microsoft 365 credentials into an operational intelligence collection and BEC fraud capability. Its distinguishing features relative to the broader criminal tooling landscape are:

- Scale: Designed to monitor dozens or hundreds of victim accounts simultaneously from a single interface
- AI integration: Shikamaru's automated analysis and conversational chat interface provide capabilities previously requiring significant manual analyst effort, enabling small criminal teams to process large account portfolios efficiently
- Operational depth: The combination of live proxied Graph access, EML offline caching, conversation bookmarking, and rich-text email composition creates a complete end-to-end BEC workflow
- Modularity: By sharing the DC Broker backend, DC-Inbox adds zero infrastructure overhead — it is a zero-cost force multiplier for operators already using DC Broker

The platform's polish - Quill.js rich text editor, multi-turn AI chat, SSE streaming research, Outlook-faithful design language - indicates significant software engineering investment and suggests an experienced development team or well-resourced criminal organization.

8.3 BoB V2

BoB V2 represents a mature, operationally efficient platform for the second phase of Microsoft 365 account takeover operations. Its key differentiators from generic phishing tools are:

- Mandatory proxy routing with integrated BitLaunch VPS provisioning — infrastructure management as a first-class feature
- Automatic inbox rule injection — persistent access established the moment a cookie is ingested
- ESP classification — intelligent targeting based on email provider for maximum delivery effectiveness
- BCC blast from victim accounts — phishing emails that pass all authentication checks because they originate from legitimate mail servers
- Full send-job lifecycle management — professional-grade campaign tooling with resume-from-checkpoint capability
- RetroBoB push-token receiver — zero-touch integration with upstream acquisition platforms

The platform represents a significant operational capability uplift for threat actors conducting large-scale BEC and internal phishing campaigns against Microsoft 365 environments.

8.4 BoB V3

BoB V3 represents a qualitative leap in criminal tooling capability. Its most significant advances over V2 are:

- Fully automated 4-stage RetroBoB pipeline with LLM company inference — approaching near-zero operator involvement in the acquisition-to-exploitation cycle
- PRO Mode multi-account campaign engine — industrial-scale phishing from compromised account pools with rotation and ESP targeting
- Bot Gate protection — active countermeasure against security vendor URL scanning
- Multi-tenant API key management — confirming BoB V3 is operated as a commercial service with external customers
- Modular architecture — significantly easier to maintain, extend, and adapt than the V2 monolith, suggesting an active development team

- BitLaunch integration — cryptocurrency-funded anonymous proxy provisioning from within the panel

BoB V3, combined with DC Broker for acquisition and DC-Inbox for intelligence, constitutes a complete, integrated, largely automated Microsoft 365 account takeover and abuse platform capable of operating at enterprise scale with minimal manual operator intervention.

9 Infrastructure

9.1 The administrative panels

App	Host/IP
DC Broker Dashboard	webdisk.xenderdriver[.]com
DC Broker Dashboard	pakclaytiles[.]com
DC Broker Dashboard	nominplanner[.]com
DC Broker Dashboard	inbox.iibrealty[.]com
DC Broker Dashboard	dc.epconta[.]com
DC Broker Dashboard	cpcalendars.ecologiaenbolivia[.]com
DC Broker Dashboard	cpcalendars.amoredellavita[.]com
DC Broker Dashboard	cdc.toursart[.]com
BoB V2 - Tier-2 Outlook	bob.khayr-misr[.]com
BoB V3 (previously BoB V2)*	bob.iibrealty[.]com

* - this host seems to be one of the most updated by the creator of tools.

9.2 The phishing websites aka “proxies”

The phishing sites that are a part of this campaign can be found via regex query `page.url:/.*\/[a-z]+\/[a-z]+\?[a-z]+=[a-f0-9]{96,}/` via URLScan.io tool or inside internal proxy logs. At the current time there’s around 4013 results, this includes some false-positive detections, however it’s rather small number.

10 MITRE ATT&CK

ATT&CK Technique	ID	DC Broker Implementation
Phishing: Spearphishing via Service	T1566.003	Device code lure delivery
Steal Application Access Token	T1528	Device code flow, token import
Valid Accounts: Cloud Accounts	T1078.004	Refresh token re-use
Email Collection	T1114.002	Inbox reader, EML sync, live fetch
Remote Email Collection	T1114.002	`/api/tokens/<id>/messages`
Email Forwarding Rule	T1114.003	Inbox rule management
Data from Cloud Storage	T1530	SharePoint/OneDrive (via session cookie)
Email Account	T1087.003	Contact enumeration
Proxy: External Proxy	T1090.002	Per-token proxy routing
Web Service (C2)	T1102	Telegram bot notifications
Automated Collection	T1119	EML sync, Shikamaru AI analysis
Business Email Compromise	T1586.002	Email send from victim account

11 Protection/Hardening

Control	Effect
Block Device Code flow in Conditional Access	Eliminates the primary acquisition vector
Compliant device requirement	Forces use of registered, managed devices — blocks PRT generation from unregistered devices
Disable legacy authentication	Reduces attack surface (though Device Code is not legacy auth)
Continuous Access Evaluation (CAE)	Enables near-real-time token revocation on policy change
Named Locations + MFA for all sign-ins	Detects and blocks anomalous proxy geography
Privileged Identity Management (PIM)	Limits blast radius if admin accounts are compromised
Enable Unified Audit Logging in Microsoft 365	Monitor for New-InboxRule events with forwarding actions
Alert on mass send volume	Any account sending >50 emails in a short window should trigger review
Monitor Graph API usage	Detect unexpected spikes in contact/message enumeration from non-standard clients
Implement Microsoft Defender for Cloud Apps policies	Detect anomalous mailbox access
Block Device Code flow in Conditional Access	Eliminates the primary upstream acquisition vector feeding RetroBoB
Enforce approved client apps	Headless Chromium should not match approved client app fingerprints
Enable Microsoft Defender for Cloud Apps anomaly detection	PRO Mode’s round-robin pattern is detectable as abnormal send volume per account
Audit “Permanent Delete” operations	Review Exchange Online audit logs for permanent deletion activity

12 Detection Opportunities

Microsoft / Exchange Online

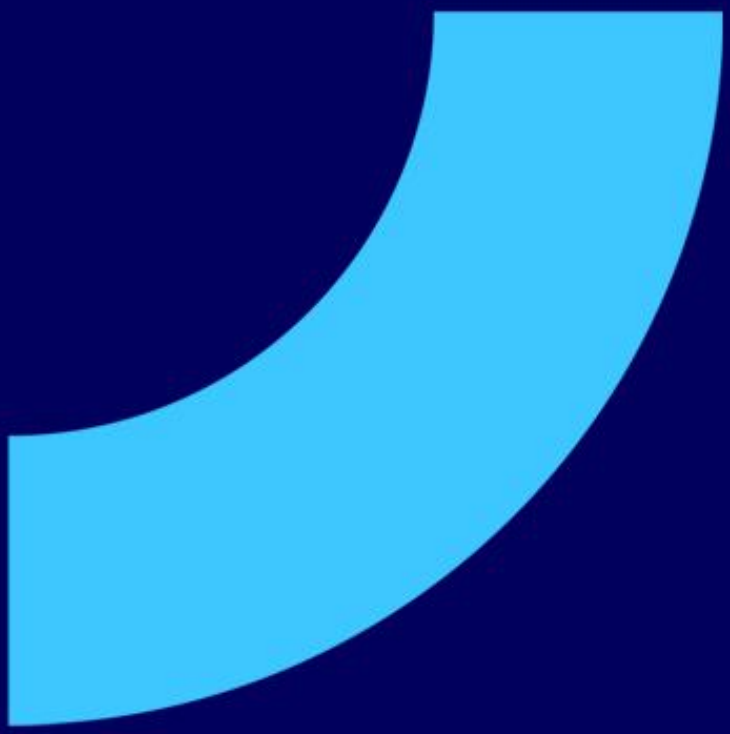
Signal	Description
Inbox rule creation from new IP	Update-InboxRule with ForwardTo or ForwardAsAttachmentTo
High-volume BCC send	Unusual send volume from account in short window
Headless browser user-agent	Sign-ins with Chrome user-agent from datacenter IPs
Contact export Graph API calls	Unusual volume of GET /contacts Graph API calls
EML batch download	Large number of GET /messages/{id}/$value calls in sequence
Multiple accounts from same proxy IP	Single IP accessing multiple different M365 tenants

About Atos Group

Atos Group is a global leader in digital transformation with c. 56,000 employees and annual revenue of c. €7.2 billion (at the go-forward perimeter), operating in 54 countries under two brands - Atos for services and Eviden for products and systems. European number one in cybersecurity and a leader in cloud, Atos Group is committed to a secure and decarbonized future and provides tailored AI-powered, end-to-end solutions for all industries. Atos Group is listed on Euronext Paris.

Find out more about us

atos.net
atos.net/career



Atos