RITS Browser v4.2.7 (Build 256) Security Architecture & Update Documentation

Root Cause Resolution, 5-Layer Defense-in-Depth, Client-Side DOM Guard & DoH Upstream Resolver

Author: RITS Security Architecture & Core Engineering Group | Date: November 11, 2026 | Document ID: REL-2026-256

Target: RITS Browser v4.2.7 (Build 256) | Package: acr.browser.ralsebrowserfull / acr.browser.ritsbrowser

Distribution Status: Production Hardening Update | Target OS: Android 7.0 - 16 (API Level 24 - 36)

1. Executive Release Overview & Root Cause Analysis

Version 4.2.7 (Build 256) represents a comprehensive security architecture upgrade for RITS Browser. This update directly resolves a critical production issue where brand lookalike warning dialogs (e.g., navigating to 'paypa1.com') failed to appear on release builds running on physical Android devices. In addressing this issue, the security subsystem was upgraded into a full 5-Layer Defense-in-Depth Phishing Architecture, providing complete client-side heuristic inspection, safe browsing interstitials, offline threat intelligence, form exfiltration guards, and upstream encrypted DoH resolver threat filtering. All 5 layers are strictly gated by the 'Extensive Security Protection' switch, enabled by default out-of-the-box.

Root Cause 1: Missing Runtime Navigation Hook in WebClient.kt

While ExtensiveSecurityEngine.kt and its unit tests existed in the :core module, ExtensiveSecurityEngine.checkUrlThreat() was never wired into the actual runtime navigation paths of WebClient.kt (neither in loadUrl() nor in CustomWebViewClient.shouldOverrideUrlLoading()). Consequently, the engine was never invoked during real omnibox submissions or hyperlink clicks.

Root Cause 2: Inverted Default Preference State (Extensive Security = false)

In AppPrefs.java, the BooleanContainer for extensive_security_mode was defaulted to false, and XML preferences specified android:defaultValue="false". On fresh installations, lookalike protection was inactive unless the user manually enabled it.

Root Cause 3: ProGuard / R8 Dead-Code Stripping in Release Builds

The app/proguard-rules.pro configuration lacked keep rules for com.ritsbrowser.core.utility.security.** and @JavascriptInterface. Because checkUrlThreat() lacked runtime entry-point callers, R8 stripped or obfuscated detection methods during release APK builds.

RESOLUTION & HARDENING SUMMARY (BUILD 256)

1. Wired checkUrlThreat() synchronously into WebClient.loadUrl() & shouldOverrideUrlLoading(), guarded by AppPrefs.extensive_security_mode.
2. Switched default value of extensive_security_mode to true across AppPrefs.java, pref_security.xml, and pref_privacy.xml.
3. Added explicit ProGuard keep rules for core security models, browser security guards, and @JavascriptInterface bridges.

2. The 5-Layer Defense-in-Depth Phishing Protection Architecture

Layer	Defense Layer Name	Implementation Class	Capabilities & Protection Scope
Layer 1	In-Memory Heuristic Threat Engine	`ExtensiveSecurityEngine.kt`	100+ monitored brands, Punycode/IDN homographs, subdomain brand spoofing (paypal.com.attacker.com), compound keywords (paypallogin.com), digraph substitutions (arnazon.com), consecutive character doubling (paypaal.com).
Layer 2	Android WebView Safe Browsing	`WebClient.kt / RitsApp.kt`	Manifest EnableSafeBrowsing meta-data, cold-start pre-warming via WebView.startSafeBrowsing() on API 27+, enforced callback.showInterstitial(true) on malicious hits.
Layer 3	Offline Threat Intelligence Engine	`ThreatIntelligenceDatabase.kt`	8KB in-memory Bloom filter (m=65536, k=4) seeded with OSINT feeds (OpenPhish, PhishTank, URLhaus), ephemeral tunnel phishing detection (ngrok, loca.lt), high-abuse TLD combinations (.top, .xyz, .club).
Layer 4	Client-Side DOM & Form Phishing Guard	`FormPhishingGuard.kt`	JavaScript DOM injection bridge, cross-domain password exfiltration detection, credential webhook harvesters (Discord/Telegram webhooks), Browser-in-the-Browser (BitB) modal overlay detection.
Layer 5	DNS-over-HTTPS (DoH) Threat Resolver	`DohThreatResolver.kt`	Encrypted asynchronous upstream resolution via Quad9 (9.9.9.9) and Cloudflare Security (1.1.1.2) with in-memory TTL caching; non-blocking zero-latency guarantee on UI thread.

3. Security Layer 1: Expanded Protected Target Corpus (100+ Brands)

In Build 256, the protected target list was expanded from 20 to over 100 tier-one targets frequently weaponized in phishing attacks. Targets are categorized by risk profile and evaluated using length-calibrated string matching (Levenshtein distance is strictly gated to brand lengths >= 5 to prevent false alarms on short words):

Banking & Financial Services	paypal, chase, bankofamerica, wellsfargo, citibank, barclays, capitalone, usbank, pnc, tdbank, schwab, fidelity, robinhood, revolut, venmo, zelle, cashapp, stripe, square, wise, hsbc, santander, bnpparibas, standardchartered, bkash, nagad, rocket, upay
Crypto & Web3 Wallets	binance, coinbase, kraken, metamask, trustwallet, opensea, kucoin, bybit, okx, ledger, trezor, gemini, bitfinex, crypto
Tech, Cloud & Enterprise	google, apple, microsoft, amazon, facebook, meta, instagram, twitter, whatsapp, telegram, spotify, netflix, dropbox, linkedin, tiktok, snapchat, reddit, discord, slack, zoom, skype, wechat, yahoo, baidu, tencent, github, gitlab, bitbucket, adobe, salesforce, atlassian, notion, figma, canva, godaddy, cloudflare, namecheap, wordpress, cpanel, oracle, cisco, intel, nvidia
E-Commerce & Retail	ebay, walmart, aliexpress, alibaba, target, bestbuy, etsy, shopify, rakuten, mercadolibre, flipkart
Streaming, Gaming & Logistics	disneyplus, hulu, youtube, twitch, steam, steampowered, epicgames, playstation, xbox, roblox, fedex, usps, dhl, ups

4. Advanced Threat Detection Modules in ExtensiveSecurityEngine.kt

Module A: Subdomain Brand Abuse Engine (checkSubdomainBrandAbuse)
Detects domains nesting a trusted brand in a subdomain (e.g. paypal.com.attacker.com or auth.paypal.phishing.org). The engine parses the host labels, identifies the registrable base domain (accounting for multi-part TLDs like .co.uk), and checks whether preceding subdomain labels contain any protected brand. Legitimate brand subdomains (e.g. login.paypal.com, pay.google.com) bypass cleanly without false positives.

Module B: Compound Keyword Spoofing Engine (findCompoundKeywordBrand)
Detects unhyphenated and hyphenated domain names pairing trusted brands with sensitive authentication/verification keywords (e.g., paypallogin.com, secure-paypal.com, amazonaccount.com, chasebanking.com). The engine searches for brand occurrences and verifies that the remaining tokens match high-risk keywords: login, signin, verify, account, banking, security, auth, portal, recovery, wallet.

Module C: Digraph Substitutions & Character Doubling (collapseConsecutiveDuplicates)
Handles visual digraph attacks where pairs of letters imitate single characters (e.g., arnazon.com where 'rn' imitates 'm', vvhatsapp.com where 'vv' imitates 'w', and 'cl' for 'd'). Additionally, an O(N) duplicate character collapse detects intentional vowel/consonant doubling designed to bypass simple distance checks (e.g. paypaal.com, gooogle.com, netflixx.com)..

Module D: Punycode & Mixed-Script IDN Homograph Detector (isIdnHomograph)
Inspects domain labels for Punycode prefixes (xn--) and executes Unicode regex matching for mixed-script attacks combining Cyrillic/Greek lookalikes with standard Latin scripts (e.g. paypal.com). Decodes Punycode into native Unicode representation for display in user warnings.

5. Security Layer 2: Android WebView Safe Browsing & Gateway Interstitials

Manifest Meta-Data Declaration
app/src/main/AndroidManifest.xml declares <meta-data android:name="android.webkit.WebView.EnableSafeBrowsing" android:value="true" /> inside the <application> tag, activating Google Safe Browsing on all supported Android WebView instances.

Early Cold-Start Pre-Warming in Application
In RitsBrowserApplication.kt (onCreate), WebView.startSafeBrowsing(this) is invoked on Android 8.1+ (API 27+). Warming up the Safe Browsing subsystem at process initialization ensures cloud threat intelligence caches are populated prior to the first page request.

Strict Interstitial Enforcement in WebClient.kt
CustomWebViewClient overrides onSafeBrowsingHit(web, request, threatType, callback). When extensive_security_mode is true, it strictly invokes callback.showInterstitial(true) to display Google's standard red warning interstitial. When disabled, it calls callback.proceed(false).

6. Security Layer 3: Offline Threat Intelligence Database & Generic Phishing Engine

ThreatIntelligenceDatabase.kt operates as an ultra-fast (< 1 µs), zero-telemetry in-memory security layer designed to catch generic, unbranded phishing sites and disposable credential traps that do not mimic a specific well-known corporate brand:

In-Memory Bloom Filter Threat Feed (m=65536, k=4)
Maintains an 8KB BitSet indexed by 4 distinct hash functions (FNV-1a and 31-multiplier polynomial rolling hash). Seeded with known malicious domains from OpenPhish, PhishTank, and URLhaus threat feeds. Provides O(1) deterministic pre-flight threat rejection with zero network overhead.

Ephemeral Dynamic Tunnel Phishing Interception
Phishing operators frequently abuse free tunneling services (ngrok.io, ngrok-free.app, loca.lt, trycloudflare.com, pagekite.me) to host temporary credential harvesters. The engine flags any dynamic tunnel hostname serving sensitive authentication or verification paths (e.g. /login, /verify).

High-Abuse TLD & Structural Token Heuristics
Detects unbranded credential-theft domains on frequently abused top-level domains (.top, .xyz, .work, .click, .buzz, .club, .cam, .cfd, .sbs, .icu) when paired with critical authentication tokens (e.g. login-credential-portal.icu, secure-login-gateway.top).

7. Security Layer 4: Client-Side DOM & Form Phishing Guard

FormPhishingGuard.kt injects an in-browser JavaScript analysis routine on page load completion (onPageCommitVisible & onPageFinished), interfacing with the native Android runtime via the registered @Javascriptinterface bridge 'RitsSecurityBridge':

Cross-Domain Password Exfiltration Interception
Inspects all <form> elements containing <input type="password">. If the form's action attribute targets an external origin distinct from the current document's host (and not a legitimate subdomain), the submission is intercepted and reported to native Android code.

Credential Webhook Harvester Detection
Phishing kits often exfiltrate harvested credentials directly to public messaging webhooks. The guard scans form action attributes and submission listeners for known webhook endpoints: discord.com/api/webhooks and api.telegram.org/bot.

Browser-in-the-Browser (BitB) Overlay Detection
Detects deceptive CSS/DOM elements simulating native browser address bars, fake SSL lock icons, or simulated popup titlebars designed to convince users they are inside a secure OAuth popup.

Native Modal Dialog Interception
When a DOM violation is detected, reportFormPhishing(reason, targetOrigin) triggers a native modal AlertDialog prompting the user: 'Credential Theft Risk Detected - This page contains a login form that exfiltrates your password to an external destination.'

8. Security Layer 5: DNS-over-HTTPS (DoH) Upstream Threat Resolver

DohThreatResolver.kt adds encrypted upstream threat filtering via Quad9 (https://dns.quad9.net/dns-query) and Cloudflare Security (https://security.cloudflare-dns.com/dns-query). Designed for strictly non-blocking asynchronous execution:

Encrypted Threat Query Resolution
Dispatches background application/dns-json queries over TLS. Quad9 synthesizes threat intelligence from 20+ cybersecurity vendors, while Cloudflare Security blocks known malware and phishing domains at the resolver level.

Zero-Latency Guarantee & NetworkOnMainThread Prevention
DoH resolution runs exclusively on Dispatchers.IO. Navigation methods in WebClient evaluate only in-memory cached results via checkHostThreatCached(). The UI thread is never blocked, and no synchronous network calls are ever made.

Sinkhole Interception & Thread-Safe Caching
Maintains a ConcurrentHashMap caching resolution status with a 5-minute TTL. If upstream intelligence returns sinkhole IPs (0.0.0.0 or 127.0.0.1), the host is cached as a threat and subsequent navigations are immediately blocked with a native warning.

9. Centralized Toggle Enforcement & Preference Synchronization

All 5 security layers are strictly gated by the master preference switch: AppPrefs.extensive_security_mode.get(). When enabled (default: true), Layer 1 heuristic evaluation, Layer 2 Safe Browsing interstitial enforcement, Layer 3 Bloom filter checks, Layer 4 DOM injection, and Layer 5 DoH resolution are fully active. When toggled OFF by the user, all extra security checks and background resolutions are completely bypassed, ensuring zero overhead and preserving total user autonomy.

10. Build System Hardening & ProGuard / R8 Configuration

ProGuard Rules Added to app/proguard-rules.pro:

```
-keep class com.ritsbrowser.core.utility.security.** { *; }
-keepclassmembers class com.ritsbrowser.core.utility.security.** { *; }
-keep class com.ritsbrowser.browser.security.** { *; }
-keepclassmembers class * { @android.webkit.JavascriptInterface <methods>; }
```

11. Comprehensive Test Suite Execution (100% Pass Rate)

Gradle Task	Result Summary	Test Coverage Details
`:core:testDebugUnitTest`	20 passed, 0 skipped, 0 failed	ExtensiveSecurityEngineTest (19 tests) + ExtensiveSecurityEngineBenchmarkTest (1 test)
`:adblock:testDebugUnitTest`	6 passed, 0 skipped, 0 failed	AdBlockerRuntimeSecurityTest + SecurityFiltersUnitTest + ExampleUnitTest
`:legacy:testDebugUnitTest`	2 passed, 0 skipped, 0 failed	VpnStatusTrackerTest + ExampleUnitTest
`:browser:compileDebugKotlin`	Build Finished Successfully	WebClient.kt navigation hooks, FormPhishingGuard wiring, and CustomWebViewClient
`:app:compileDebugKotlin`	Build Finished Successfully	RitsBrowserApplication SafeBrowsing warmup and manifest compilation

12. Empirical Verification Matrix Across All 5 Defense Layers

Test Vector	Layer	Engine Category	Detection Mechanism	Observed Behavior
`https://www.paypal.com/webscr`	Layer 1	`TYPOSQUATTING`	Normalized 1 -> l character substitution	PROMPT: Warning Dialog intercepted (Passed)
`https://arnazon.com/deals`	Layer 1	`TYPOSQUATTING`	Digraph rn -> m visual normalization	PROMPT: Warning Dialog intercepted (Passed)
`https://paypaal.com/signin`	Layer 1	`TYPOSQUATTING`	Consecutive duplicate 'a' collapsed	PROMPT: Warning Dialog intercepted (Passed)
`https://paypallogin.com/auth`	Layer 1	`COMPOUND_KEYWORD`	Protected brand + auth keyword 'login'	PROMPT: Warning Dialog intercepted (Passed)
`https://paypal.com.attacker.com`	Layer 1	`SUBDOMAIN_BRAND`	Subdomain label brand traversal	PROMPT: Warning Dialog intercepted (Passed)
`testsafebrowsing.appspot.com`	Layer 2	`SAFE_BROWSING`	Android WebView Safe Browsing interstitial	INTERSTITIAL: Red Warning Displayed (Passed)
`https://fast-verify-id.com`	Layer 3	`KNOWN_MALICIOUS`	8KB Bloom filter (m=65536, k=4) hit	PROMPT: Warning Dialog intercepted (Passed)
`https://sub.ngrok-free.app/login`	Layer 3	`EPHEMERAL_TUNNEL`	Ephemeral dynamic tunnel + login path	PROMPT: Warning Dialog intercepted (Passed)
`https://auth-update.top/billing`	Layer 3	`GENERIC_PHISHING`	High-abuse .top TLD + auth tokens	PROMPT: Warning Dialog intercepted (Passed)
`Form posting password to external host`	Layer 4	`FORM_EXFILTRATION`	JavaScript DOM cross-origin form check	PROMPT: Modal Warning Dialog intercepted (Passed)
`Form posting to Discord webhook`	Layer 4	`FORM_EXFILTRATION`	Webhook credential harvester URL pattern	PROMPT: Modal Warning Dialog intercepted (Passed)
`https://login.paypal.com`	Safe	`NONE (Legitimate)`	Base domain matches protected brand	PASSED: Normal Page Load (No interruption)
`https://www.google.com/search`	Safe	`NONE (Legitimate)`	Exact single-token brand match	PASSED: Normal Page Load (No interruption)