INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-28 06:33 UTC

Robinhood Onboarding HTML Injection Abused to Deliver Auth-Passing Phishing from Official Domain

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0229
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Robinhood trading platform (onboarding email pipeline, Device field input handling); Gmail users (dot-aliasing behavior exploited for account creation bypass)
Published	2026-04-27T19:11:01
Discovery Source	Rss

Executive Summary

Threat actors exploited an HTML injection flaw in Robinhood's account onboarding pipeline to send phishing emails that originated from noreply@robinhood.com, passing all standard email authentication checks. Attackers precision-targeted victims using the 7 million email addresses stolen in Robinhood's 2021 data breach, making the campaign exceptionally credible to recipients. The business risk extends beyond Robinhood: any organization with unsanitized input fields in transactional email templates is potentially vulnerable to the same trusted-sender abuse pattern.

Technical Analysis

Attackers injected malicious HTML into an unsanitized Device field within Robinhood's account onboarding email template. Because the injected content was rendered inside emails dispatched by Robinhood's own sending infrastructure, the resulting messages passed SPF and DKIM authentication - standard email authentication controls produced no alert signal. No CVE has been publicly assigned. Relevant weaknesses are CWE-79 (Improper Neutralization of Input During Web Page/Email Generation) and CWE-284 (Improper Access Control, unauthenticated abuse of the transactional email relay). MITRE ATT&CK coverage includes T1566.002 (Spearphishing Link), T1078 (Valid Accounts, trusted sending domain abuse), and T1589.002/T1114 (breach data reuse for targeting). Gmail dot-aliasing (e.g., j.ohn.doe@gmail.com vs johndoe@gmail.com) was exploited to generate high volumes of account registrations from a limited set of real addresses, industrializing onboarding flow abuse. Robinhood remediated by removing the unsanitized Device field. No CVSS vector string or EPSS

data has been published for this flaw. The 2021 breach dataset (7 million records) was attributed to pompompurin; the April 2026 phishing campaign operator remains unattributed.

Action Checklist

1. Audit your transactional email templates immediately: identify every field that accepts user-supplied input and confirm it is sanitized before rendering in email body or headers. Treat any unsanitized field as an active risk pending remediation.
2. Query email gateway and SIEM logs for outbound transactional emails containing unexpected HTML tags or JavaScript in body content. Flag any account-creation or onboarding email where the Device, User-Agent, or similar metadata fields contain angle brackets, script tags, or encoded equivalents. Review for anomalous registration volume patterns (high-frequency signups from Gmail dot-alias variants).
3. Remove or strictly sanitize all user-controlled input fields that flow into email templates. Apply output encoding (HTML entity encoding) to every variable rendered in email content. Enforce an allowlist for Device field values if the field is functionally constrained.
4. After sanitization controls are deployed, send test registrations with HTML injection payloads and confirm the rendered email contains only escaped text. Monitor transactional email bounce and complaint rates for anomalies. Verify SPF/DKIM records remain intact and have not been modified.
5. Review the full scope of user-controlled inputs across all transactional email pipelines (password reset, notification, invoice, onboarding). Implement Content Security Policy headers where applicable. Evaluate whether your organization's breach history (any prior data exposure) could supply a precision targeting list for a similar campaign - if so, treat affected users as elevated-risk recipients and consider proactive notification.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to legal and executive leadership if analysis confirms that injected emails were delivered to recipients whose addresses appear in your organization's prior breach disclosure records, as this constitutes a second-order harm to previously affected individuals and may trigger mandatory breach notification timelines under applicable state or federal law.
Recovery Notes	After deploying HTML entity encoding across all transactional email templates, run a minimum of 48 hours of post-fix monitoring on ESP complaint and bounce rates before closing the incident, as residual phishing emails already delivered to recipient inboxes may continue generating user confusion and credential submission for days beyond containment. Verify that all accounts created during the campaign window via Gmail dot-alias bypass are flagged for manual review and that any associated sessions or access tokens issued to those accounts are invalidated. Continue monitoring your registration endpoint for renewed injection attempts for at least 30 days post-fix, as threat actors who successfully exploited this vector are likely to probe for regressions or alternative unsanitized fields.

Forensic Artifacts	ESP raw MIME message archives for all onboarding emails sent during the campaign window — these preserve the rendered email body exactly as delivered, showing whether Device or User-Agent field content appeared as executable HTML and passed DKIM/SPF signing by noreply@robinhood.com (or your equivalent sending domain), confirming the authentication bypass mechanism | Web application registration endpoint logs (Apache/Nginx access logs or application-layer request logs) containing the full POST body or at minimum the Device and User-Agent field values submitted during account creation — injection strings (, , data: URIs) in these fields are direct evidence of attacker tooling and technique | Account registration database export filtered on accounts created in burst periods from gmail.com addresses, with dot-normalized local-parts — duplicate canonical addresses map the scope of the Gmail dot-alias bypass used to exhaust your per-address registration limits and evade deduplication controls | ESP delivery and complaint feedback loop (FBL) reports for the noreply sending address during the campaign window — spam complaint spikes against your own authenticated sending domain indicate recipients recognized the phishing content despite DKIM/SPF passing, and these reports establish victim harm for any regulatory notification analysis | Git or deployment history for the email template rendering codebase, specifically commits touching the Device, User-Agent, or metadata field handling functions — absence of escaping in the commit history establishes when the vulnerable code was introduced and how long the attack surface existed prior to exploitation

Per-Action IR Details

Containment — Audit your transactional email templates immediately: identify every field that accepts user-supplied input and confirm it is sanitized before rendering in email body or headers. Treat any unsanitized field as an active risk pending remediation.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST SI-10 (Information Input Validation), NIST SI-2 (Flaw Remediation), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure)

Compensating: For a 2-person team without enterprise tooling: grep -rn --include='*.html' --include='*.mjml' --include='*.hbs' --include='*.jinja' '{{.*device\|.*user.agent\|.*metadata' ./email-templates/ to locate unescaped variable injection points in template files. Simultaneously pull your ESP (SendGrid, Mailgun, SES) template API and diff all template versions from the past 90 days using the provider's audit log export — most free-tier accounts retain this. Flag any template variable that renders a string value without a |escape or autoescape wrapper.

Evidence: Before touching templates, export and preserve: (1) your ESP's full template render logs or preview cache showing the exact rendered output of any onboarding/account-creation email sent within the past 30 days — these will show whether Device/User-Agent field content appeared literally in email body HTML; (2) your account registration database records for any signup where the Device or User-Agent field contains angle brackets (), %3C/%3E URL-encoded equivalents, or HTML entity sequences (<, >) — these rows are direct evidence of injection attempts; (3) a static export of all current transactional email template source files with file hashes (sha256sum *.html > template_hashes_$(date +%Y%m%d).txt) before any remediation changes alter them.

Detection — Query email gateway and SIEM logs for outbound transactional emails containing unexpected HTML tags or JavaScript in body content. Flag any account-creation or onboarding email where the Device, User-Agent, or similar metadata fields contain angle brackets, script tags, or encoded equivalents. Review for anomalous registration volume patterns (high-frequency signups from Gmail dot-alias variants).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, run these queries directly against your ESP's log export (CSV/JSON) using jq or Python pandas: (1) jq '.[] | select(.subject | test("Welcome|Verify|Confirm")) | select(.body | test("]|%3C|%3E|<|>|<|>' registration_db_export.csv | grep -i 'device\|user.agent' to isolate injection-bearing registration records. Cross-reference registration timestamps with your ESP's outbound send log to confirm whether injected content reached recipient inboxes.

Evidence: Preserve before analysis: (1) Your ESP's full outbound message log for all onboarding/account-creation sends over the past 90 days, including the raw MIME source of flagged messages — the raw MIME will show whether injected HTML rendered in the email body or was stripped; (2) Web application access logs (Apache/Nginx) for your registration endpoint, filtered on POST requests to /signup or /register, retaining the full request body or at minimum the User-Agent and any device metadata fields — look for payloads containing 50 registrations/hour from gmail.com addresses) and normalize dots out of the local-part to identify how many map to the same canonical address, which reveals the scope of the bypass campaign against the Robinhood 2021 breach list.

Eradication — Remove or strictly sanitize all user-controlled input fields that flow into email templates. Apply output encoding (HTML entity encoding) to every variable rendered in email content. Enforce an allowlist for Device field values if the field is functionally constrained.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-10 (Information Input Validation), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams without a WAF or enterprise input-validation library: (1) Implement Python's html.escape() or equivalent (PHP htmlspecialchars($var, ENT_QUOTES, 'UTF-8'), Node.js he.encode()) on every template variable before it is passed to the email rendering function — this is a code-level fix requiring no tooling budget; (2) For the Device field specifically, if values are expected to be OS/browser strings, enforce a server-side regex allowlist: ^(Windows|macOS|iOS|Android|Linux)[\w\s\.\-]{0,50}$ and reject or strip non-matching values before they touch the template engine; (3) Commit all sanitization changes with a signed git commit and tag the release (git tag -s eradication-htmlinject-YYYYMMDD) so you have a tamper-evident audit trail of when the fix was deployed, satisfying NIST AU-10 (Non-Repudiation) requirements without additional tooling.

Evidence: Before deploying the fix, capture: (1) A code diff export (git diff HEAD~30 -- '**/email*' '**/template*' '**/mailer*' > pre_fix_template_diff.patch) showing the exact lines where user-supplied variables were rendered without escaping — this is your root-cause evidence; (2) A sample of raw MIME exports from your ESP showing at least one confirmed injected email that passed DKIM/SPF intact, proving the authentication bypass mechanism worked via Robinhood's own signing infrastructure rather than domain spoofing; (3) Database rows where Device field content contains injection strings, exported with timestamps and associated account IDs, to determine whether the injected registrations were created by automated tooling (uniform timing intervals) or human operators (irregular timing), which informs threat actor attribution.

Recovery — After sanitization controls are deployed, send test registrations with HTML injection payloads and confirm the rendered email contains only escaped text. Monitor transactional email bounce and complaint rates for anomalies. Verify SPF/DKIM records remain intact and have not been modified.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-6 (Security and Privacy Function Verification), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST IR-4 (Incident Handling), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For a 2-person team: (1) Regression test using a controlled registration with payload Device value of alert(1) and > — receive the resulting onboarding email and view raw source to confirm the output renders as <script>alert(1)</script> and not executable HTML; (2) Query your DNS registrar or use dig TXT example.com to verify your SPF record and compare against a previously documented baseline hash — any modification to the SPF include chain or DKIM selector could indicate an attacker attempted to expand mail-sending authority; (3) Use MXToolbox

(free) or mail-tester.com to send a post-fix test onboarding email and confirm DKIM=pass, SPF=pass, DMARC=pass with no alignment failures — screenshot and timestamp this result as evidence of restored integrity.

Evidence: Preserve before declaring recovery complete: (1) Side-by-side raw MIME captures of a pre-fix injected onboarding email (from ESP archive) and a post-fix test email with the same payload, demonstrating that escaping is now applied — this is your remediation proof artifact; (2) ESP bounce and complaint rate baseline for onboarding emails from the 30 days prior to the campaign, compared against rates during the campaign window — a spike in spam complaints against noreply@robinhood.com-originated messages during the attack period is forensic evidence of recipient harm; (3) A timestamped dig +dnssec TXT _dmarc.yourdomain.com && dig +dnssec TXT yourdomain.com output confirming DMARC policy, SPF record, and DKIM key selectors are unchanged from pre-incident configuration.

Post-Incident — Review the full scope of user-controlled inputs across all transactional email pipelines (password reset, notification, invoice, onboarding). Implement Content Security Policy headers where applicable. Evaluate whether your organization's breach history (any prior data exposure) could supply a precision targeting list for a similar campaign — if so, treat affected users as elevated-risk recipients and consider proactive notification.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST RA-3 (Risk Assessment), NIST SI-10 (Information Input Validation), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 3.2 (Establish and Maintain a Data Inventory)

Compensating: For a 2-person team: (1) Build a transactional email input inventory in a spreadsheet: for each email type (password reset, invoice, notification, onboarding), list every template variable, its data source (user-submitted vs. system-generated), and whether it passes through an escape function — any user-submitted variable without a confirmed escape function is a finding; (2) Use semgrep with the community ruleset (semgrep --config=p/owasp-top-ten .) against your email rendering codebase to automatically flag unescaped template variables — free, runs locally, no telemetry required; (3) Cross-reference your user database against known breach datasets using Have I Been Pwned's domain search API (free for organization owners) to identify what percentage of your user base appears in prior breach compilations — this quantifies the precision-targeting risk specific to your organization, mirroring exactly how the Robinhood 2021 breach list was weaponized in this campaign.

Evidence: Capture for the post-incident lessons-learned record: (1) A complete inventory of all transactional email pipelines with variable-level sanitization status — this documents the full attack surface that existed, not just the one exploited field; (2) Historical registration records normalized to detect any Gmail dot-alias account clusters that were created before this campaign was detected, which may reveal an earlier reconnaissance or staging phase; (3) Any threat intelligence correlation between your known-exposed user email addresses (from prior breach events) and the accounts targeted or created during this campaign — if the same addresses appear in both sets, it confirms the attacker used a breach-derived targeting list, a finding that may trigger breach notification obligations under CCPA, GDPR, or state notification laws depending on jurisdiction and data classification.

Detection Guidance

For organizations reviewing their own email pipelines: parse outbound transactional email logs for body content containing HTML tags originating from user-supplied fields. In your email sending platform (SendGrid, SES, Postmark, or equivalent), enable logging of template variable values at render time and alert on values matching patterns such as , script, img src, or URL-encoded equivalents (%3C, %3E). For phishing detection on the receiving side: this campaign is specific to Robinhood infrastructure. Look for inbound emails purportedly from noreply@robinhood.com that pass SPF/DKIM but contain unexpected link destinations (hover-check URLs that do not resolve to robinhood.com properties). Behavioral indicator: high-velocity account creation requests from Gmail addresses that differ only by dot placement (e.g., multiple registrations resolving to the same canonical address). No public IOCs (hashes, IPs, sender infrastructure specific to the April 2026 campaign) have been

reported by Robinhood, threat intelligence vendors, or public security research as of publication.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`noreply@robinhood.com (sending address)`	Legitimate Robinhood sending address abused via HTML injection in onboarding pipeline; not itself malicious but used as the authenticated sender for phishing messages	HIGH

Framework Mappings

MITRE-ATTACK

- **T1566.001** — Spearphishing Attachment
- **T1598** — Phishing for Information
- **T1585.001** — Social Media Accounts
- **T1586.002** — Email Accounts
- **T1566.002** — Spearphishing Link
- **T1589.002** — Email Addresses
- **T1114** — Email Collection
- **T1078** — Valid Accounts

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A03:2021** — Injection

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **16.10** — Apply Secure Design Principles in Application Architectures
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC7.4** — Responds to identified security incidents

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(5)(i)** — Security Awareness and Training
- **164.308(a)(6)(ii)** — Response and Reporting

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information

NIST-CSF-2

- **RS.CO-03** — Recovery activities and progress communicated

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1566.001	Spearphishing Attachment	Initial-Access
T1598	Phishing for Information	Reconnaissance
T1585.001	Social Media Accounts	Resource-Development
T1586.002	Email Accounts	Resource-Development
T1566.002	Spearphishing Link	Initial-Access
T1589.002	Email Addresses	Reconnaissance
T1114	Email Collection	Collection
T1078	Valid Accounts	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/robinhood-account-cr...	T3
	https://www.bleepingcomputer.com/news/security/robinhood-account-cr...	T3
	https://www.bleepingcomputer.com/news/security/microsoft-365-direct...	T3
	https://www.bleepingcomputer.com/news/security/7-million-robinhood-...	T3
Robinhood Stock Could Suffer After Users Report Phishing Incident	https://finance.yahoo.com/markets/stocks/articles/robinhood-stock-c...	T3

Generated 2026-04-28 06:33 UTC by TJS Security Command Center