INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-09 14:22 UTC

# FROST: Browser-Based SSD Timing Attack Fingerprints Sites and Apps Without Permissions or Native Code

SECURITY ANALYSIS | HIGH | CVSS 7.5

| | |
|---|---|
| SCC Item ID | SCC-STY-2026-0180 |
| Type | Security Analysis |
| Severity | HIGH |
| CVSS Base Score | 7.5 |
| Affected Products | Chrome (Google), Firefox (Mozilla), Safari (Apple), all major desktop browsers on macOS and Linux; Origin Private File System (OPFS) API |
| Published | 2026-06-09T05:50:41 |
| Discovery Source | Rss |

## Executive Summary

Researchers at Graz University of Technology demonstrated FROST, a JavaScript-only browser attack that infers which websites a user visits and which desktop applications they have open, all from an ordinary web page, with no permissions required. The attack exploits shared SSD storage queues measurable through the browser's Origin Private File System API, affecting Chrome, Firefox, and Safari on macOS and Linux. All three major vendors were notified and none has shipped a mitigation, leaving every organization whose employees browse the web exposed to passive behavioral surveillance from any webpage they visit.

## Technical Analysis

FROST (the name given by Graz University of Technology researchers) is a browser-based side-channel attack that exploits storage I/O contention to infer user behavior without any privileges, browser extensions, native code, or user-granted permissions. The attack surface is the Origin Private File System (OPFS) API, a browser primitive designed to give web applications fast, sandboxed file storage. The critical insight is that OPFS operations share the physical SSD's I/O queue with all other processes on the machine, including the operating system, other browser tabs, and native applications. Because SSD request queues are a shared resource, contention patterns from a victim's legitimate activity produce measurable timing delays in an attacker-controlled page's own OPFS operations.

The attack mechanics map to two well-established weakness classes: CWE-208 (Information Exposure Through Timing Discrepancy) and CWE-203 (Observable Behavioral Discrepancy). The attacker's JavaScript issues a stream of storage operations and measures how long each takes. When the victim loads a known website or opens a known application, that activity competes for the same SSD queue, altering the timing profile the attacker observes. By correlating observed timing signatures against a pre-built fingerprint library of known sites and applications, the attacker can infer what the victim is doing, silently, continuously, and from any open browser tab.

The MITRE ATT&CK mapping is instructive: T1592 (Gather Victim Host Information) and T1592.004 (Client Configurations) describe the reconnaissance objective; T1185 (Browser Session Hijacking, used here in the broader sense of passive browser-based surveillance) and T1189 (Drive-by Compromise, as the delivery mechanism requires only that the victim visit or have open a malicious or compromised page) frame the delivery model. The CVSS 7.5 (High) rating reflects a network-delivered attack with low complexity and no prerequisites beyond an active tab, a realistic deployment threshold for any threat actor willing to buy ad inventory, compromise a high-traffic site, or serve a malicious iframe.

The defensive gap this research exposes is architectural: browsers grant JavaScript timing access to a shared hardware resource without rate limiting, jitter, or resolution reduction sufficient to defeat side-channel inference. Previous browser-based timing attacks (Spectre, DRAMA, various cache-timing variants) prompted vendors to reduce timer resolution and add jitter to APIs like performance.now(). FROST demonstrates that OPFS provides a new, high-resolution timing channel that existing mitigations do not cover. As of the paper's publication date, no vendor has shipped a fix. The research was conducted under coordinated disclosure with Google, Mozilla, and Apple notified prior to publication, the absence of patches at publication is notable and signals the difficulty of mitigating a fundamental shared-resource problem without degrading legitimate OPFS use cases.

The privacy implications extend beyond individual browsing history. An attacker who can fingerprint which applications a user has open, accounting software, VPN clients, security tools, communication platforms, gains operational intelligence useful for targeted social engineering, timing attacks on authentication workflows, or identifying high-value targets within a browsing population.

## Action Checklist

1. Step 1: Assess exposure, every organization whose employees use Chrome, Firefox, or Safari on macOS or Linux desktops is exposed; this is not a niche enterprise product. Confirm browser fleet composition and OS mix through your asset inventory (CIS Control 1.1: Establish and Maintain Detailed Enterprise Asset Inventory).
2. Step 2: Review controls, enforce Content Security Policy (CSP) headers on all organization-controlled web properties to limit cross-origin script execution; audit whether internal web apps use or expose the OPFS API unnecessarily, and consider disabling OPFS access via enterprise browser policy where the API is not required for legitimate use.
3. Step 3: Update threat model, add browser-based storage timing side-channels as a passive reconnaissance vector in your threat register. Map to T1592 (Gather Victim Host Information) and T1189 (Drive-by Compromise). Note that delivery requires no user interaction beyond visiting a page, malicious ad networks and compromised third-party scripts are viable delivery paths.
4. Step 4: Communicate findings, brief leadership that this is a passive surveillance capability, not a data-exfiltration breach. The immediate risk is behavioral fingerprinting of employees visiting sensitive internal portals or using specific security tools, context that matters for insider threat, targeted phishing,

and nation-state reconnaissance scenarios.

5. Step 5: Monitor developments, track browser vendor security advisories (Chrome Security, Mozilla Security Advisories, Apple Security Updates) for OPFS-specific mitigations. No vendor has patched as of the Graz publication date; expect mitigations to involve timer jitter or API rate-limiting. Subscribe to vendor advisory feeds and re-evaluate enterprise browser policy when patches ship.

6. Step 6: Evaluate browser isolation controls, where feasible, review whether enterprise browser isolation or remote browser isolation (RBI) solutions in your environment limit the attack surface by separating browser execution from the local hardware context.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | URGENT |
| **Escalation Criteria** | Escalate to CISO and legal if evidence emerges that employee browsing behavior on sensitive internal portals (HR, finance, security tooling) has been exposed to a third-party adversary via FROST-style fingerprinting, particularly if the organization operates under HIPAA, GDPR, or SOC 2 obligations where behavioral surveillance of employees accessing regulated data could trigger breach notification or privacy impact assessment requirements. |
| **Recovery Notes** | There is no patch to apply as of the Graz publication date, so recovery is posture-based: verify that CSP headers restricting cross-origin script execution are confirmed active on all internal web properties and that OPFS API access is disabled via enterprise browser policy on macOS and Linux endpoints where not operationally required. Monitor Chrome, Firefox, and Safari vendor advisory channels continuously until each vendor ships an OPFS-specific mitigation (expected to involve high-resolution timer jitter or OPFS rate-limiting); re-evaluate enterprise browser policy within 72 hours of each patch release to confirm the specific mitigation mechanism addresses the SSD queue timing vector. Maintain the advisory watch log for a minimum of 180 days given that all three vendors had not shipped fixes at disclosure time. |

| | |
|---|---|
| **Forensic Artifacts** | Browser OPFS storage directories on macOS (~/Library/Application Support/Google/Chrome/Default/File System/ and equivalent Firefox profile paths under ~/Library/Application Support/Firefox/Profiles/) and on Linux (~/.config/google-chrome/Default/File System/, ~/.mozilla/firefox//storage/default/) — an attacker running FROST writes and reads temporary files through these paths; timestamps and file creation events in these directories during a suspicious page visit are direct evidence of OPFS API abuse. \| Web server and reverse-proxy access logs for internal applications (Apache access_log, nginx access.log, or equivalent) — look for requests to internal portal URIs from macOS/Linux browser User-Agents immediately preceded by a referrer from an unknown or ad-network domain, indicating a potential FROST delivery chain where the malicious page fingerprinted the user before they navigated to the sensitive resource. \| DNS resolver query logs (Pi-hole query log at /var/log/pihole.log, pfSense DNS resolver log, or enterprise DNS logs) — FROST delivery via compromised third-party scripts or malvertising would produce DNS lookups to ad-network or CDN domains from affected workstations; correlate these queries by timestamp with OPFS directory activity on the same host to reconstruct the delivery chain. \| macOS Unified Log entries (`log show --predicate 'subsystem == "com.apple.webkit"' --info`) and Linux kernel audit log entries (`ausearch -k file_system -ts today`) capturing file operations within the browser OPFS storage paths — these provide timestamped evidence of OPFS read/write activity that would be generated by a JavaScript timing loop exploiting the shared SSD queue. \| Browser process execution telemetry from Sysmon (Event ID 1, Process Create) on Linux endpoints or Endpoint Security Framework events on macOS — specifically, child process creation or unusual I/O patterns spawned by the browser renderer process (chrome --renderer, firefox --content-process) during visits to untrusted pages, which can help bound the time window of a potential FROST fingerprinting session. |

### Per-Action IR Details

**Step 1: Assess exposure — every organization whose employees use Chrome, Firefox, or Safari on macOS or Linux desktops is exposed; this is not a niche enterprise product. Confirm browser fleet composition and OS mix through your asset inventory (CIS 1.1: Establish and Maintain Detailed Enterprise Asset Inventory).**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: Establish IR capability and inventory assets prior to incident declaration

**Controls:** CIS 1.1 (IG1/IG2/IG3) — Establish and Maintain Detailed Enterprise Asset Inventory, CIS 2.1 (IG1/IG2/IG3) — Establish and Maintain a Software Inventory, CIS 2.2 (IG1/IG2/IG3) — Ensure Authorized Software is Currently Supported

**Compensating:** Run `osquery` with query `SELECT name, version, type FROM programs WHERE name LIKE '%Chrome%' OR name LIKE '%Firefox%' OR name LIKE '%Safari%';` on macOS/Linux endpoints to enumerate installed browser versions fleet-wide without EDR. On Linux, supplement with `dpkg -l | grep -E 'chromium|firefox'` or `rpm -qa | grep -E 'chromium|firefox'` via a simple SSH loop or Ansible ad-hoc command across managed hosts.

**Evidence:** Before scoping, export current software inventory snapshots (osquery `SELECT * FROM apps;` on macOS; package manager listings on Linux) to establish a baseline of browser versions present — relevant because FROST exploits OPFS API availability, which is tied to specific browser versions across Chrome, Firefox, and Safari on macOS and Linux only (Windows is not affected per Graz research).

**Step 2: Review controls — enforce Content Security Policy (CSP) headers on all organization-controlled web properties to limit cross-origin script execution; audit whether internal web apps use or expose the OPFS API unnecessarily, and consider disabling OPFS access via enterprise browser policy where the API is not required for legitimate use.**

**NIST Phase:** Containment

**Reference:** NIST 800-61r3 §3.3 — Containment Strategy: Select and implement containment measures to limit adversary capability while preserving operations

**Controls:** AC-4 — Information Flow Enforcement, CIS 4.6 (IG1/IG2/IG3) — Securely Manage Enterprise Assets and Software

**Compensating:** For CSP enforcement without a WAF appliance: add `Content-Security-Policy: default-src 'self'; script-src 'self'` response headers directly in Apache (`Header always set Content-Security-Policy`) or nginx (`add_header Content-Security-Policy`) configs for all internal web apps. To audit OPFS usage across internal apps, grep application source or web server access logs for `navigator.storage` or `createSyncAccessHandle` string patterns: `grep -r 'createSyncAccessHandle\|navigator\.storage\.getDirectory' /var/www/` — a 2-person team can complete this review in a single sprint. For Chrome enterprise fleet, deploy the `DefaultFileSystemReadGuardSetting` / `FileSystemWriteAskForUrls` group policy values via a JSON managed preferences file under `/etc/opt/chrome/policies/managed/` on Linux.

**Evidence:** Capture current web server response headers for all internal applications before modifying configs: `curl -I https://internalapp.example.com` — document which internal apps lack CSP headers or include permissive `script-src *` values, as these represent FROST delivery surfaces. Also extract Chrome enterprise policy state via `chrome://policy` export or `defaults read com.google.Chrome` on macOS before changes, to confirm pre-remediation OPFS API exposure.

**Step 3: Update threat model — add browser-based storage timing side-channels as a passive reconnaissance vector in your threat register. Map to T1592 (Gather Victim Host Information) and T1189 (Drive-by Compromise). Note that delivery requires no user interaction beyond visiting a page — malicious ad networks and compromised third-party scripts are viable delivery paths.**

**NIST Phase:** Post Incident

**Reference:** NIST 800-61r3 §4 — Post-Incident Activity: Use lessons learned to update threat models, detection capabilities, and policies

**Controls:** CIS 7.1 (IG1/IG2/IG3) — Establish and Maintain a Vulnerability Management Process

**Compensating:** Document FROST as a threat register entry referencing MITRE ATT&CK T1592.001 (Hardware) and T1189 (Drive-by Compromise); note that no exploit code execution or permission prompt is involved — the attack is entirely passive JavaScript timing measurement via the OPFS `createSyncAccessHandle` API. Use a free threat modeling tool such as OWASP Threat Dragon or a structured markdown register in a git repo to capture the delivery path (malicious ad network JS → OPFS timing loop → SSD queue inference) as the attack chain narrative.

**Evidence:** Before finalizing the threat model update, collect any existing DNS query logs or proxy logs showing employee browser traffic to known ad-network domains (e.g., from Pi-hole or squid proxy logs) — these establish realistic delivery path exposure. Also document the browser version-to-OPFS-capability matrix from vendor changelogs to record when OPFS became available in each browser, establishing the historical exposure window.

**Step 4: Communicate findings — brief leadership that this is a passive surveillance capability, not a data-exfiltration breach. The immediate risk is behavioral fingerprinting of employees visiting sensitive internal portals or using specific security tools — context that matters for insider threat, targeted phishing, and nation-state reconnaissance scenarios.**

**NIST Phase:** Detection Analysis

**Reference:** NIST 800-61r3 §3.2 — Detection and Analysis: Estimate impact and scope; provide findings to authorized staff

**Controls:** AU-6 — Audit Record Review, Analysis, And Reporting

**Compensating:** Produce a one-page executive brief structured around three FROST-specific risk scenarios: (1) an adversary fingerprinting which employees access the VPN portal or internal HR system from an employee's browser during a malvertising campaign; (2) nation-state reconnaissance identifying use of specific security tooling (e.g., Burp Suite, Wireshark) by detecting the associated processes via SSD timing; (3) targeted spear-phishing enabled by correlating browsing behavior inferred from a compromised third-party site. No SIEM required — the brief can be drafted from the Graz University research paper findings alone.

**Evidence:** Before briefing, pull proxy or DNS resolver logs (e.g., from pfSense DNS resolver logs or Pi-hole query logs) for the past 30 days to identify which employees regularly visit sensitive internal portals from browsers on macOS/Linux — this establishes the concrete population at risk for behavioral fingerprinting and makes the leadership brief specific rather than theoretical.

**Step 5: Monitor developments — track browser vendor security advisories (Chrome Security, Mozilla Security Advisories, Apple Security Updates) for OPFS-specific mitigations. No vendor has patched as of the Graz publication date; expect mitigations to involve timer jitter or API rate-limiting. Subscribe to vendor advisory feeds and re-evaluate enterprise browser policy when patches ship.**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: Maintain situational awareness through ongoing monitoring of threat and vulnerability intelligence

**Controls:** CIS 7.1 (IG1/IG2/IG3) — Establish and Maintain a Vulnerability Management Process, CIS 7.2 (IG1/IG2/IG3) — Establish and Maintain a Remediation Process, CIS 7.3 (IG1/IG2/IG3) — Perform Automated Operating System Patch Management, CIS 7.4 (IG1/IG2/IG3) — Perform Automated Application Patch Management

**Compensating:** Set up free RSS or Atom feed monitoring for the three vendor advisory streams: Chrome Releases blog (https://chromereleases.googleblog.com), Mozilla Security Advisories (https://www.mozilla.org/en-US/security/advisories/), and Apple Security Updates (https://support.apple.com/en-us/111900) using a self-hosted RSS aggregator such as FreshRSS or a simple cron job with `curl` piped to `grep -i 'OPFS\|file system\|timing\|storage'`. When a patch ships, use a Sigma rule (free, YAML-based) targeting Chrome/Firefox update event log entries to confirm deployment across the fleet before closing the advisory watch.

**Evidence:** Maintain a running advisory watch log (a dated plaintext or markdown file is sufficient) recording each vendor's public position on FROST/OPFS timing mitigations — relevant because the absence of a patch is itself a material fact for risk acceptance documentation and any future regulatory inquiry about whether the organization tracked the vulnerability lifecycle.

**Step 6: Evaluate browser isolation controls — where feasible, review whether enterprise browser isolation or remote browser isolation (RBI) solutions in your environment limit the attack surface by separating browser execution from the local hardware context.**

**NIST Phase:** Containment

**Reference:** NIST 800-61r3 §3.3 — Containment Strategy: Implement controls that reduce adversary capability to exploit the vulnerable component

**Controls:** AC-3 — Access Enforcement, AC-4 — Information Flow Enforcement, CIS 4.4 (IG1/IG2/IG3) — Implement and Manage a Firewall on Servers, CIS 4.5 (IG1/IG2/IG3) — Implement and Manage a Firewall on End-User Devices

**Compensating:** For teams without a commercial RBI budget: deploy Chrome or Firefox inside a Linux container (e.g., Docker with `--device` restrictions and no `/dev/nvme*` passthrough) for high-risk browsing sessions involving sensitive internal portals. A 2-person team can script a `docker run --rm -it --security-opt seccomp=chrome.json jlesage/firefox` containerized browser profile for analyst workstations within a day. This directly addresses FROST's exploitation mechanism — the attack measures SSD queue timing through the OPFS API; container storage isolation removes the shared hardware queue the attack depends on.

**Evidence:** Before evaluating isolation solutions, document the current browser execution context for high-risk roles (SOC analysts, executives, finance): are browsers running on bare metal with direct NVMe access, in VMs, or already in containers? Capture `lsblk` and `mount` output on representative macOS/Linux endpoints to confirm whether OPFS writes would reach physical SSD queues — this determines whether existing virtualization already partially mitigates FROST or whether the full attack surface is exposed.

## Detection Guidance

FROST is a passive timing side-channel, from the victim's perspective, no malicious payload is delivered, no process is spawned, and no file is written to disk. Standard endpoint detection signatures will not fire. Detection must focus on the attacker's infrastructure and on policy anomalies rather than endpoint telemetry.

Network and DNS logging (aligned with NIST AU-2: Event Logging and AU-12: Audit Record Generation): Look for browser connections to pages that make unusually high-frequency, low-payload OPFS API calls. In practice this is difficult to distinguish from legitimate use without browser-level instrumentation, but anomalous JavaScript execution patterns making rapid, repetitive storage operations with no visible user-facing function may warrant scrutiny in environments with browser telemetry.

Content Security Policy audit: Review CSP header implementation across organization-controlled web properties. Absent or permissive CSP allows third-party scripts (ad networks, analytics, CDN-hosted libraries) to execute OPFS operations. Log and alert on CSP violations (AU-6: Audit Record Review, Analysis, and Reporting).

Third-party script inventory: Enumerate all third-party JavaScript loaded by internal and public-facing web applications. Unrecognized or recently added scripts from ad networks or analytics providers are candidate delivery vectors. This is a supply-chain hygiene check, not a runtime detection.

Hunting hypothesis: If your environment has browser telemetry (e.g., enterprise browser, endpoint agent with browser visibility), hunt for tabs that maintain long-lived OPFS file handles with no corresponding user-facing storage function. A page that opens an OPFS file and issues thousands of small reads/writes per minute with no user-visible application purpose is a behavioral anomaly worth investigating.

No specific IOCs (hashes, domains, IPs) were published in the source research at the time of this writing. The attack technique is proof-of-concept; no active threat actor campaigns using FROST have been attributed. Detection posture should focus on the technique class rather than specific indicators.

## Indicators of Compromise

| Type | Value | Context | Confidence |
|---|---|---|---|
| TOOL | `Pending — refer to Graz University of Technology FROST research paper for published proof-of-concept details` | Proof-of-concept JavaScript using OPFS API timing measurements to fingerprint visited sites and open applications; no active campaign IOCs published at time of story | LOW |

## Framework Mappings

### MITRE-ATTACK

- **T1185** — Browser Session Hijacking
- **T1592** — Gather Victim Host Information
- **T1189** — Drive-by Compromise
- **T1592.004** — Client Configurations

### NIST-800-53R5

- **SC-7** — Boundary Protection

### CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

### ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

### SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1185 | Browser Session Hijacking | Collection |
| T1592 | Gather Victim Host Information | Reconnaissance |
| T1189 | Drive-by Compromise | Initial-Access |
| T1592.004 | Client Configurations | Reconnaissance |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://thehackernews.com/2026/06/new-frost-attack-lets-websites-tr... | T3 |
| **Origin private file system - Web APIs \| MDN** | https://developer.mozilla.org/en-US/docs/Web/API/File_System_API/Or... | T3 |
| **Origin private file system \| Hacker News** | https://news.ycombinator.com/item?id=42137790 | T3 |
| **The origin private file system \| Articles - web.dev** | https://web.dev/articles/origin-private-file-system | T3 |
| **Intent to prototype & ship: Origin Private File System API (OPFS)** | https://groups.google.com/a/mozilla.org/g/dev-platform/c/dsRxP4liTek | T3 |

Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-09 14:22 UTC by TJS Security Command Center