

Gemini Drift: Automated Patch-Twin Discovery for NASA Flight Software via Semantic Graph Analysis

Bee Rosa Davis
 NASA Johnson Space Center
 Flight Operations Directorate (FOD)
 2101 NASA Parkway, Houston, TX 77058
 bee.r.davis@nasa.gov
 bee_davis@alumni.brown.edu

Abstract—We present *Gemini Drift*, a system for automatically discovering unpatched sibling vulnerabilities (“drift variants”) in NASA’s Core Flight System (cFS) by detecting when fixes applied to one module have not propagated to structurally or semantically similar code elsewhere. Gemini Drift combines (i) a revised patch-similarity score, (ii) positive-unlabeled learning for sibling likelihood, (iii) neighborhood risk propagation over call graphs, and (iv) SAVES, a bounded semantic micro-slicing checker that verifies whether the patch’s safety invariants (e.g., bounds, MsgID/length checks, watchdog timing) are enforced in candidate siblings. A mission-criticality gate tightens thresholds for human-rated and flagship missions, while heritage-aware boosting prioritizes code reused across programs. A retrospective study design on cFS (6.7→6.8) targets precision@10 ≈ 0.45 and sub-6 hour turn-around with non-weaponized evidence packs suitable for Flight Software Board review.

Index Terms—vulnerability discovery, patch analysis, flight software, cFS, semantic verification, graph propagation, positive-unlabeled learning

I. INTRODUCTION

NASA’s cFS underpins dozens of missions. Patches that fix a defect in one app often fail to propagate to semantically similar code elsewhere, leaving “drift variants.” Manual patch-diffing by experts is effective but does not scale across 50+ applications and mission variants. Gemini Drift systematizes *post-patch* variant hunting: rank likely twins, verify the patch’s intent via semantic invariants, and surface review-ready evidence packs.

Goals: (1) mission-aware safety (tighter gates for human-rated), (2) semantic correctness (verify invariants, not just diffs), (3) scalability (bounded analysis budgets), and (4) operational fitness (non-weaponized outputs and FSB-ready artifacts).

II. RELATED WORK

Patch diffing and variant analysis (e.g., BinDiff/Diaphora), graph-based program similarity, and PU learning for weak labels inform our approach. Semantic equivalence checking and bounded symbolic execution provide rigor at the function level. Flight-software-specific literature emphasizes watchdogs, message authentication, and table integrity; we embed those concerns as invariants in SAVES. (Full references omitted here for brevity.)

III. METHOD

A. Notation

Let $G = (V, E, W)$ be the directed call graph; $u \in V$ a function; $(u \rightarrow v) \in E$; $W(u, v) \geq 0$ a normalized edge weight. We set

$$W(u, v) = \frac{\text{call_count}(u \rightarrow v)}{\sum_w \text{call_count}(u \rightarrow w)}, \quad (\text{III.1})$$

falling back to a static-likelihood heuristic when counts are unavailable. $N_k(u)$ is the k -hop neighborhood (default $k=2$). Each function has a base embedding $x_{\text{base}}(u) \in \mathbb{R}^d$ (IR/graph features; $d=128$). A patched neighbor is p , with pre-patch representation $x_{\text{base}}(p^-)$. Let $\text{sink}(u)$ denote cFS sink labels visible in u (SB payload copies, table loads, PSP mem ops).

B. Revised Patch Similarity PSI'

We combine structural similarity, sink overlap, and a semantic term from SAVES:

$$\begin{aligned} \text{PSI}'(u; p) = & \lambda_c \cos(x_{\text{base}}(u), x_{\text{base}}(p^-)) \\ & + \lambda_s \frac{|\text{sink}(u) \cap \text{sink}(p^-)|}{|\text{sink}(u) \cup \text{sink}(p^-)|} + \lambda_g \mathcal{G}(u; p), \end{aligned} \quad (\text{III.2})$$

where

$$\mathcal{G}(u; p) = \mathbb{I}[\text{Verdict}(u; p) \in \{\text{mismatch}, \text{partial}\}] \quad (\text{III.3})$$

(from SAVES, Sec. III-D). Weights satisfy $\lambda_c + \lambda_s + \lambda_g = 1$.

$$\begin{aligned} \hat{R}(\theta) = & \pi_P \mathbb{E}_{x \sim P}[\ell(g_\theta(x))] \\ & + \max\left\{0, \mathbb{E}_{x \sim U}[\ell(-g_\theta(x))] - \pi_P \mathbb{E}_{x \sim P}[\ell(-g_\theta(x))]\right\}. \end{aligned} \quad (\text{III.4})$$

Uncertainty via an M -member ensemble:

$$\overline{\text{USL}}(u) = \frac{1}{M} \sum_{m=1}^M \text{USL}_m(u), \quad (\text{III.5})$$

$$\sigma^2(u) = \frac{1}{M-1} \sum_{m=1}^M (\text{USL}_m(u) - \overline{\text{USL}}(u))^2. \quad (\text{III.6})$$

a) *Calibration.*: We apply temperature scaling to minimize ECE:

$$\text{USL}_{\text{cal}}(u) = \sigma\left(\frac{1}{T} \logit(\text{USL}(u))\right), \quad (\text{III.7})$$

$$\text{ECE} = \sum_{b=1}^B \frac{|B_b|}{n} |\text{acc}(B_b) - \text{conf}(B_b)|. \quad (\text{III.8})$$

C. Neighborhood Risk Propagation SRI

Let $d(u, v)$ be hop distance and $\delta \in (0, 1)$ a decay. With $\varepsilon = 10^{-8}$ for stability,

$$\begin{aligned} \text{SRI}_{\text{raw}}(u) &= (1 - \alpha) \text{USL}_{\text{cal}}(u) \\ &+ \alpha \frac{\sum_{v \in N_k(u)} \delta^{d(u,v)} W(u, v) \text{USL}_{\text{cal}}(v)}{\varepsilon + \sum_{v \in N_k(u)} \delta^{d(u,v)} W(u, v)}. \end{aligned} \quad (\text{III.9})$$

Heritage boost $H(u) \in [0, 1]$ (normalized log mission count):

$$H(u) = \frac{\log(1 + \text{missions}(u))}{\log(1 + \max_m \text{missions}(m))}. \quad (\text{III.10})$$

Final risk:

$$\text{SRI}(u) = \text{SRI}_{\text{raw}}(u) (1 + \eta H(u)). \quad (\text{III.11})$$

Significance gate. We assess the heritage effect via a label-permutation test with $n=1000$ iterations at $\alpha=0.05$. The test achieves $\approx 80\%$ power to detect standardized effect sizes $d>0.3$ at our pilot sample sizes. We enable $\eta>0$ only when the test is significant; otherwise set $\eta=0$.

D. Semantic Parity Checker (SAVES)

Let $\text{Pred}(f)$ be SMT predicates gathered on bounded slices (depth ≤ 3) from sinks to inputs. For patched p , new invariants

$$\Delta = \text{Post}(p) \setminus \text{Pre}(p)$$

are extracted from before/after predicate sets (bounds checks, MsgID/length checks, watchdog timing, table validation, legal state transitions). Verdict:

$$\text{Verdict}(u; p) = \begin{cases} \text{match} & \text{Pred}(u) \models \Delta, \\ \text{partial} & \exists \Delta' \subset \Delta : \text{Pred}(u) \models \Delta', \\ \text{mismatch} & \text{Pred}(u) \not\models \Delta, \\ \text{inconclusive} & \text{mapping ambiguous or timeboxed.} \end{cases} \quad (\text{III.12})$$

Path prioritization and selection. For a slice path π , define a normalized priority

$$\widehat{\text{prio}}(\pi) = \frac{w^\top \phi(\pi)}{\sum_i |w_i|}, \quad (\text{III.13})$$

where $\phi(\pi)$ indicates SB-tainted sizes, memcpy/str* motifs, missing MsgID/length checks, absence/presence of table validation, watchdog gating, etc. Let $\Pi(u)$ be all slice paths; we select the top set

$$\Pi_{100}(u) = \arg \max_{\Pi' \subseteq \Pi(u), |\Pi'| \leq 100} \sum_{\pi \in \Pi'} \widehat{\text{prio}}(\pi),$$

breaking ties by *shortest taint distance* to the sink (fewest edges from source to sink along π). Only $\Pi_{100}(u)$ is submitted to SAVES within runtime budgets.

E. Mission-Criticality Gates and Drift Monitoring

Escalate when

$$\begin{aligned} \text{USL}_{\text{cal}}(u) &\geq \tau_{\text{USL}} + a_{\text{mission}}, \\ \text{SRI}(u) &\geq \tau_{\text{SRI}} + a_{\text{mission}}, \\ \sigma(u) &\leq \sigma_{\text{max}}. \end{aligned} \quad (\text{III.14})$$

Addends: human-rated 0.05, flagship 0.04, robotic 0.02, Cube-Sat 0.00.

F. Runtime Budgets and Complexity

Budgets: $t_{\text{per-path}} \leq 3$ s, $t_{\text{per-func}} \leq 30$ s, $t_{\text{global-SAVES}} \leq 30$ min. Complexity (per cycle): embeddings $O(|V|d)$; scoring $O(|V|)$; SAVES $O(k \cdot b^d)$ with $k \leq 10$, $b \sim 5$, $d \leq 3$.

IV. SYSTEM ARCHITECTURE

Pipeline. (1) Build-specific preprocessing and symbolization per target configuration; (2) embedding and feature extraction; (3) scoring (PSI', USL, SRI); (4) SAVES verification on top- k ; (5) evidence pack assembly (non-weaponized: no exploit code); (6) telemetry/decision logging.

Call graph fidelity. Direct calls from compiler output; indirect targets via (i) function-pointer tables in .data, (ii) type-compatible signatures when tables are unavailable, and (iii) conservative inclusion of app callbacks. This over-approximates possible edges; empirical target: $>95\%$ edge coverage for direct calls and $>80\%$ for indirect in typical cFS apps. False positives are acceptable (advisory) versus silent misses.

V. EVALUATION DESIGN

Retrospective cFS 6.7→6.8. Analyze ~ 40 – 50 patches across 10–12 apps; construct ground-truth sibling sets via manual review.

Metrics. Precision@10, recall of known variants, ECE after calibration, solver timeout rates, wall-clock (fetch→evidence pack < 6 h), and memory (host: ~ 32 GB RAM; embeddings ~ 2 GB; solver peak ~ 8 GB).

Expected performance. Top-10 precision ≈ 0.45 for unseen patches (conservative; retrospective pilot observed ≈ 0.70 on known sets), with 2–4 FSB-worthy evidence packs/month.

VI. DISCUSSION AND LIMITATIONS

Preprocessor variants. We run post-preprocessing per target and *do not* naively merge conflicts; configuration-specific findings keep their macros/flags in metadata. Cross-config drift is flagged, not collapsed.

SAVES timeouts. Inconclusive results (e.g., solver timeout) are *auto-escalated* for mandatory manual review within 48 hours. For human-rated missions, escalations bypass normal triage and go directly to the mission software lead.

False positives. Over-approximate graphs and broad taint can inflate findings. We manage burden by (1) ranking by confidence, (2) including concrete code patterns in evidence packs, and (3) retuning thresholds using FSB disposition history.

Out of scope (v0.1). ISRs and deep concurrency races; Byzantine-fault tolerance. These are v0.2 directions.

VII. OPERATIONAL INTEGRATION (CONOPS)

Advisory integration: Gemini Drift runs post-merge on patch branches and during monthly release prep. It does not block builds; instead, it reduces review load via pre-triaged candidates. Human-rated escalations produce out-of-band advisories (30-day window) to the Flight Software Board and mission leads. Based on historical patch rates (~ 15 /quarter), we estimate 2–4 evidence packs/month requiring FSB review.

VIII. CONCLUSION

Gemini Drift automates post-patch twin discovery for cFS by combining similarity, PU learning, graph risk propagation, and bounded semantic verification. Mission-aware gates and heritage significance testing focus attention where it matters most. The system delivers non-weaponized, FSB-ready evidence with bounded runtime—turning patch discipline into a measurable, repeatable process.