

Event-Driven Integration Blueprint

ServiceNow + GitHub Secret Protection & Code Security

Dependabot, Code Scanning, Secret Scanning, and
Security Campaign Automation

Prepared for Internal AppSec / SecOps Enablement
Architecture & Engineering Reference

Version 2.0 — Updated: January 14, 2026

Abstract

This document provides a comprehensive architectural blueprint for integrating GitHub’s security products (Secret Protection and Code Security, formerly bundled as GitHub Advanced Security) with ServiceNow’s Vulnerability Response and Application Vulnerability Response modules. The integration enables automated intake, triage, assignment, SLA enforcement, and remediation tracking for Dependabot alerts, Code Scanning alerts, and Secret Scanning alerts through event-driven webhook processing, REST API reconciliation, and native ServiceNow integration patterns.

Key Updates (2025–2026): This revision incorporates GitHub’s April 2025 product unbundling into GitHub Secret Protection and GitHub Code Security, the November 2025 GA release of alert assignees and security campaigns with corresponding webhook and REST API support, enhanced Copilot Autofix integration, and updated ServiceNow GitHub Application Vulnerability Integration capabilities.

Contents

1	Executive Summary	4
1.1	Recommended Strategy: Hybrid Architecture	4
2	Target Outcomes and Non-Functional Requirements	5
2.1	Operational Outcomes	5
2.2	Non-Functional Requirements (NFRs)	5
3	Reference Architecture	6
3.1	High-Level Component Diagram	6
3.2	Data Flow: Event Path (Near Real-Time)	7
3.3	Data Flow: Baseline Path (Authoritative Ingestion)	8
4	Integration Patterns and Selection Criteria	8
4.1	Pattern A: OOTB Import-Driven (Scheduled)	8
4.2	Pattern B: Webhook-First Event Processing	9
4.3	Pattern C: Webhooks + Queue + Reconciliation (Recommended)	9

5	Data Model Mapping	10
5.1	ServiceNow Domain Objects	10
5.2	Canonical Alert Identifier	10
5.3	State Mapping Matrix	10
5.4	Alert Assignee Mapping (New 2025)	11
6	GitHub Configuration	11
6.1	Webhook Events to Enable	11
6.2	Webhook Security Controls	12
6.3	GitHub App Authentication (Recommended)	12
6.4	REST APIs for Reconciliation	13
7	ServiceNow Configuration	13
7.1	Baseline: GitHub Application Vulnerability Integration	13
7.2	Event-Driven Inbound Options	13
7.2.1	Option 1: External Receiver (Recommended)	13
7.2.2	Option 2: ServiceNow Scripted REST API	14
7.2.3	Option 3: IntegrationHub with External Trigger	14
7.3	Recommended ServiceNow Workflow Components	14
8	Implementation: Webhook Receiver	15
8.1	Node.js/TypeScript Implementation	15
8.2	Go Implementation	20
8.3	ServiceNow Scripted REST API Implementation	26
9	Implementation: Reconciliation Worker	31
9.1	Python Reconciliation Service	31
10	Implementation: GitHub Actions Integration	43
10.1	Security Results to ServiceNow DevOps	43
11	Automation Design: Playbooks and Workflows	46
11.1	Triage and Prioritization Matrix	46
11.2	Assignment Resolution Strategy	47
11.3	SLA Configuration Matrix	47
11.4	Exception Management Workflow	47
12	Reliability Engineering	47
12.1	Idempotency Implementation	47
12.2	Retry and Dead-Letter Strategy	48
12.3	Observability Requirements	48
13	Security Hardening Checklist	49
14	Rollout Plan	50
14.1	Phase 0: Prerequisites (Week 1–2)	50
14.2	Phase 1: Baseline Import (Week 3–4)	50
14.3	Phase 2: Webhook Event Path (Week 5–8)	50
14.4	Phase 3: Reconciliation and Advanced Features (Week 9–12)	50

14.5 Phase 4: Production Hardening (Week 13–16)	51
15 Appendix A: Webhook Payload Examples	51
15.1 Code Scanning Alert Created (with Assignees)	51
15.2 Secret Scanning Alert with Push Protection Bypass	52
16 Appendix B: Normalized Event Envelope	53
17 Appendix C: Primary References	54
17.1 GitHub Documentation	54
17.2 GitHub Changelog (2025)	54
17.3 ServiceNow Documentation	54

1 Executive Summary

An event-driven system integrating ServiceNow and GitHub's security products can be implemented to automate the complete vulnerability management lifecycle for:

- **Dependabot alerts** — Software Composition Analysis (SCA) covering dependency vulnerabilities, with automated PR generation and auto-merge capabilities for low-risk updates
- **Code Scanning alerts** — Static Application Security Testing (SAST) powered by CodeQL and third-party SARIF-producing tools, now with GA Copilot Autofix suggestions and alert assignees
- **Secret Scanning alerts** — Credential exposure detection including push protection bypass events, AI-powered detection, validity checking, and security campaigns for coordinated remediation

GitHub provides first-class webhook events for these alert families (`dependabot_alert`, `code_scanning_alert`, `secret_scanning_alert`) and mandates validating deliveries using the `X-Hub-Signature-256` HMAC header.¹

Note

Product Evolution (April 2025): GitHub Advanced Security (GHAS) has been unbundled into two standalone products:

- **GitHub Secret Protection** (\$19/month per active committer) — push protection, secret scanning, AI-powered detection, validity checking
- **GitHub Code Security** (\$30/month per active committer) — code scanning, Copilot Autofix, security campaigns, Dependency Review Action

Both products are now available to GitHub Team plan customers without requiring GitHub Enterprise.^a

^a<https://github.blog/changelog/2025-03-04-introducing-github-secret-protection-and-github-code-security/>

On the ServiceNow side, the **GitHub Application Vulnerability Integration** (via the ServiceNow Store) provides the most direct path to import findings into **Vulnerability Response** / **Application Vulnerability Response** and manage remediation workflows at scale.²

1.1 Recommended Strategy: Hybrid Architecture

A production-grade design employs a **hybrid** approach combining three complementary integration patterns:

1. **Baseline Ingestion** — Official ServiceNow integration application for authoritative normalization, deduplication, GHSA-to-CVE mapping, and SecOps domain model alignment
2. **Event-Driven Updates** — GitHub webhooks into a hardened receiver (API Gateway, cloud function, or ServiceNow Scripted REST API) for immediate routing, assignment, security campaign synchronization, and workflow triggers

¹<https://docs.github.com/en/webhooks/using-webhooks/validating-webhook-deliveries>

²<https://store.servicenow.com/store/app/006d4fe21b646a50a85b16db234bcba2>

3. **Reconciliation Polling** — Scheduled jobs using GitHub REST APIs to detect missed events, synchronize assignee changes, campaign status updates, and ensure state convergence

This approach preserves OOTB integration benefits while achieving near-real-time automation with provable correctness.

2 Target Outcomes and Non-Functional Requirements

2.1 Operational Outcomes

- **Time-to-triage reduction:** Automated classification, routing, and prioritization at ingestion with sub-minute latency for critical alerts
- **SLA enforcement:** Deterministic due dates based on severity, exploitability (EPSS), asset criticality, and alert type (secrets typically warrant shortest SLAs)
- **Automated lifecycle tracking:** State synchronization (open → in-progress → fixed/dismissed) bidirectionally between GitHub and ServiceNow
- **Security campaign orchestration:** Synchronized campaign status, deadlines, and assignee notifications across both platforms
- **Audit-grade traceability:** Immutable event log of alert state transitions, assignment changes, and workflow actions with correlation IDs
- **Copilot Autofix integration:** Track autofix suggestions, acceptance rates, and time-to-remediation metrics

2.2 Non-Functional Requirements (NFRs)

Category	Requirements
Security	<ul style="list-style-type: none"> • HMAC-SHA256 signature verification for all webhook deliveries • Least-privilege GitHub App tokens with fine-grained permissions • Secrets vaulting (never log secret values from secret scanning) • OAuth 2.0 / mutual TLS for ServiceNow API authentication • IP allow-listing and WAF protection for webhook endpoints
Reliability	<ul style="list-style-type: none"> • Idempotent processing using <code>X-GitHub-Delivery</code> and canonical alert keys • Exponential backoff retries with jitter • Dead-letter queues (DLQ) with alerting and manual replay capability • At-least-once delivery guarantee with deduplication

Scalability

- Burst handling for webhook storms (large codebase scans)
- Queue-based backpressure with configurable concurrency
- GitHub API rate limit awareness (5,000 requests/hour for Apps)
- Horizontal scaling of webhook processors

Maintainability

- Clear ownership boundaries: GitHub config / integration code / ServiceNow workflows
 - Version-controlled mapping rules and transformation logic
 - Staged rollout capability with canary repositories
 - Comprehensive logging and distributed tracing
-

3 Reference Architecture

3.1 High-Level Component Diagram

The integration architecture consists of five primary layers:

1. Source Layer (GitHub)

- GitHub Secret Protection (secret scanning, push protection)
- GitHub Code Security (CodeQL, code scanning, Copilot Autofix)
- Dependabot (dependency vulnerability alerts, automated PRs)
- Security Campaigns (coordinated remediation initiatives)

2. Transport Layer (Webhooks + APIs)

- Outbound webhooks with HMAC signatures
- REST API v3 for reconciliation and bidirectional sync
- GraphQL API for complex queries and bulk operations

3. Ingestion Layer (Webhook Receiver)

- Signature validation and request authentication
- Event normalization and enrichment
- Durable queue publication (SQS, Pub/Sub, Azure Service Bus)
- DLQ handling and replay infrastructure

4. Processing Layer (Integration Services)

- Event transformation and mapping
- ServiceNow API client with retry logic
- Reconciliation scheduler

- Metrics and observability collection

5. Destination Layer (ServiceNow)

- Scripted REST API or IntegrationHub endpoint
- Vulnerability Response / Application Vulnerability Response
- Flow Designer workflows for automation
- CMDB integration for asset correlation

3.2 Data Flow: Event Path (Near Real-Time)

1. GitHub emits a webhook delivery on alert lifecycle changes:

- **Code Scanning:** created, reopened, closed_by_user, fixed, appeared_in_branch, reopened_by_user
- **Secret Scanning:** created, resolved, revoked, reopened, validated (validity status change)
- **Dependabot:** created, dismissed, fixed, reintroduced, auto_dismissed, auto_reopened
- **New (2025):** Alert assignee changes trigger webhook events for both code scanning and secret scanning alerts³

2. The **Webhook Receiver** validates:

- HMAC signature via **X-Hub-Signature-256** header
- Event type via **X-GitHub-Event** header
- Unique delivery ID via **X-GitHub-Delivery** header
- Optional: IP allow-list against GitHub's webhook IP ranges

3. Receiver publishes normalized event to durable queue with metadata:

- Original payload (optionally redacted for secrets)
- Computed canonical alert key
- Received timestamp and processing priority

4. ServiceNow consumer processes the event:

- Idempotent upsert into staging/event table
- Map to Vulnerability Response domain objects
- Trigger Flow Designer subflows for triage, assignment, and SLA
- Update GitHub alert state/assignee if bidirectional sync enabled

³<https://github.blog/changelog/2025-12-16-code-scanning-alert-assignees-are-now-generally-available/>

3.3 Data Flow: Baseline Path (Authoritative Ingestion)

The ServiceNow **GitHub Application Vulnerability Integration** provides:

- **Code Scanning Integration:** SAST data import with rule metadata, CWE mapping, and CVSS enrichment
- **Dependabot Integration:** SCA data with advisory severity, GHSA-to-CVE correlation, and remediation guidance
- **Secret Scanning Integration:** Credential exposure findings with secret type, validity status, and location metadata
- **Recent Improvements (2025):** Enhanced GHSA ID fallback when CVE is absent, improved delta sync for Dependabot alerts using sort field parameters⁴

Best Practice

Hybrid Approach Rationale: GitHub's integration guidance recommends webhooks for long-term metric collection and real-time alerting, while REST APIs provide point-in-time snapshots requiring polling infrastructure. The ServiceNow integration app provides optimal alignment to SecOps data models. Combining both yields speed *and* correctness.

4 Integration Patterns and Selection Criteria

4.1 Pattern A: OOTB Import-Driven (Scheduled)

Use when: Fastest time-to-value is priority and minutes-to-hours latency is acceptable for non-critical alerts.

Mechanism: Configure the GitHub Application Vulnerability Integration to ingest findings on a scheduled cadence (hourly, daily).

Strengths:

- Normalized ingestion with GHSA/CVE deduplication
- Native SecOps workflows (assignment, exception handling, false positives)
- Minimal custom code; vendor-supported upgrade path
- Automatic CWE and CVSS enrichment

Limitations:

- Not truly event-driven; insufficient for rapid-response use cases
- Secret exposure may remain unaddressed for hours
- No support for real-time assignee or campaign synchronization

⁴<https://store.servicenow.com/store/app/006dafe21b646a50a85b16db234bcba2>

4.2 Pattern B: Webhook-First Event Processing

Use when: Near-real-time automation is required (seconds to minutes latency).

Mechanism: Configure organization or repository webhooks for security alert events:

- `code_scanning_alert`
- `dependabot_alert`
- `secret_scanning_alert`
- `security_and_analysis` (for enablement/disablement tracking)

Strengths:

- Immediate alerting, incident creation, and paging/escalation
- Avoids API polling overhead and rate limit consumption
- Supports real-time assignee and campaign synchronization
- Can trigger lightweight workflows without waiting for import cadence

Limitations:

- Requires idempotency and replay design
- Possible missed deliveries without reconciliation
- More complex infrastructure requirements

4.3 Pattern C: Webhooks + Queue + Reconciliation (Recommended)

Use when: Near-real-time processing *and* provable correctness are both required.

Mechanism:

- Webhooks for instant triggers with acknowledgment within 10 seconds
- Durable message queue (SQS, Pub/Sub, Service Bus) with DLQ
- Periodic REST reconciliation for convergence (every 15–60 minutes)
- Security campaign status sync via REST API

Warning

For **secret scanning alerts**, always err on the side of immediate action. Leaked credentials can be exploited within minutes of exposure. Pattern C is strongly recommended for any deployment handling secret scanning.

5 Data Model Mapping

5.1 ServiceNow Domain Objects

ServiceNow Vulnerability Response uses the following core tables:

- **Vulnerable Items** (`sn_vul_vulnerable_item`): Infrastructure-style vulnerabilities tied to CIs
- **Application Vulnerable Items** (`sn_vul_app_vulnerable_item`): Application-scoped findings from third-party scanners
- **Third-Party Vulnerabilities** (`sn_vul_third_party_entry`): External vulnerability definitions (CVE, GHSA, CWE)
- **Vulnerability Groups**: Logical groupings for bulk operations and reporting

The GitHub Application Vulnerability Integration maps findings to Application Vulnerable Items with proper third-party entry linkage.

5.2 Canonical Alert Identifier

Define a deterministic, unique identifier for idempotent upserts:

```

1 {
2   "format": "<provider>:<org>/<repo>:<alert_type>:<alert_number>",
3   "examples": [
4     "github:acme/payments:dependabot:1234",
5     "github:acme/api-gateway:code_scanning:5678",
6     "github:acme/infrastructure:secret_scanning:9012"
7   ]
8 }
```

Listing 1: Canonical Alert Key Format

Store this key in:

- Custom event staging table (`x_ghas_event`)
- External reference field on the target SecOps record
- Correlation ID for bidirectional synchronization

5.3 State Mapping Matrix

GitHub Status	ServiceNow State	Automation Actions
open / active	Open / New	Create or reopen task; start SLA clock; assign to owner
dismissed	Risk Accepted / False Positive / Exception	Require documented reason; risk acceptance owner approval; set expiration date; record compensating controls

fixed / resolved	Resolved		Validate via PR merge, dependency update, or secret rotation; stop SLA clock; update metrics
reopened	Reopened		Re-trigger assignment; recalculate SLA; notify original assignee
auto_dismissed	Auto-Closed (Dependency)	(Dependency)	Record auto-dismissal reason; no manual action required
reintroduced	Reopened (Regression)		High-priority flag; escalate if repeat occurrence

5.4 Alert Assignee Mapping (New 2025)

With alert assignees now GA for both code scanning and secret scanning:

```

1  {
2    "alert_key": "github:acme/api:code_scanning:1234",
3    "assignees": [
4      {
5        "github_login": "jsmith",
6        "servicenow_user": "john.smith@acme.com",
7        "assigned_at": "2026-01-14T10:30:00Z",
8        "assigned_by": "security-bot"
9      }
10   ],
11   "sync_direction": "github_to_servicenow",
12   "last_synced": "2026-01-14T10:30:05Z"
13 }
```

Listing 2: Assignee Synchronization Payload Structure

6 GitHub Configuration

6.1 Webhook Events to Enable

At minimum, enable these webhook events at the organization level:

```

1  webhook_events:
2    # Core security alert events
3    - code_scanning_alert      # SAST findings lifecycle
4    - dependabot_alert        # SCA/dependency vulnerabilities
5    - secret_scanning_alert    # Credential exposure detection
6
7    # Supplementary events (recommended)
8    - security_and_analysis    # Feature enablement changes
9    - repository_advisory      # Private vulnerability reports
10
11   # CI/CD correlation (optional)
12   - check_run                 # CodeQL analysis completion
13   - workflow_run              # Security workflow status
```

Listing 3: Required Webhook Events

6.2 Webhook Security Controls

1. **Configure webhook secret:** Generate a cryptographically strong secret (minimum 32 characters):

```
1 openssl rand -base64 32
2
```

Listing 4: Generate Webhook Secret

2. **Validate X-Hub-Signature-256:** GitHub computes HMAC-SHA256 of the payload body using your secret
3. **Log X-GitHub-Delivery:** Unique UUID for each delivery; use for deduplication and replay tracking
4. **Verify X-GitHub-Event:** Ensure event type matches expected values to prevent injection
5. **Optional IP filtering:** GitHub publishes webhook IP ranges via the Meta API

6.3 GitHub App Authentication (Recommended)

For production integrations, use a GitHub App instead of personal access tokens:

```
1 github_app_permissions:
2   repository_permissions:
3     # Read access for alert data
4     security_events: read
5     contents: read
6     metadata: read
7
8     # Write access for bidirectional sync
9     security_events: write # To update alert state/assignees
10
11   organization_permissions:
12     # For org-wide alert access
13     organization_administration: read
14
15   webhook_events:
16     - code_scanning_alert
17     - dependabot_alert
18     - secret_scanning_alert
19     - security_and_analysis
```

Listing 5: GitHub App Required Permissions

Benefits of GitHub App:

- Higher rate limits (5,000 requests/hour vs 5,000/hour for PATs)
- Fine-grained permissions scoped to specific repositories
- Installation tokens with short TTL (1 hour)
- Better audit trail and revocation capabilities
- No dependency on individual user accounts

6.4 REST APIs for Reconciliation

These endpoints support periodic snapshots and repair:

Alert Type	API Endpoint
Code Scanning	GET /repos/{owner}/{repo}/code-scanning/alerts GET /orgs/{org}/code-scanning/alerts
Secret Scanning	GET /repos/{owner}/{repo}/secret-scanning/alerts GET /orgs/{org}/secret-scanning/alerts
Dependabot	GET /repos/{owner}/{repo}/dependabot/alerts GET /orgs/{org}/dependabot/alerts
Security Campaigns	GET /orgs/{org}/security-campaigns (New 2025)

7 ServiceNow Configuration

7.1 Baseline: GitHub Application Vulnerability Integration

Install and configure from the ServiceNow Store:

1. Navigate to **System Applications** → **All Available Applications** → **All**
2. Search for “GitHub Application Vulnerability Integration”
3. Install the application (requires SecOps license)
4. Configure credentials:
 - Create a Connection record with GitHub App credentials
 - Configure organization scope and repository filters
 - Set import schedule (recommended: hourly for critical repos)

7.2 Event-Driven Inbound Options

7.2.1 Option 1: External Receiver (Recommended)

Deploy a hardened webhook receiver outside ServiceNow:

- **Infrastructure:** API Gateway (AWS, Azure, GCP) + serverless function
- **Responsibilities:** Signature validation, rate limiting, queue publication
- **Benefits:** Enhanced security, independent scaling, multi-region support
- **ServiceNow integration:** Authenticated REST call with OAuth bearer token

7.2.2 Option 2: ServiceNow Scripted REST API

Direct webhook endpoint within ServiceNow:

- **Use case:** Simpler deployments, moderate volumes
- **Requirements:** Strict ACLs, signature verification, async processing
- **Limitation:** ServiceNow must respond within webhook timeout

7.2.3 Option 3: IntegrationHub with External Trigger

Leverage ServiceNow's IntegrationHub:

- Create External Trigger Definition tied to Flow Designer flow
- Configure authentication (OAuth, API Key, mutual TLS)
- Use GitHub Spoke for bidirectional management where available

7.3 Recommended ServiceNow Workflow Components

```
1  flow_designer_subflows:
2    # Inbound Processing
3    - name: ProcessGHASWebhookEvent
4      trigger: External Trigger / Scripted REST
5      actions:
6        - ValidatePayload
7        - ComputeCanonicalKey
8        - UpsertStagingRecord
9        - QueueForProcessing
10
11   # Triage and Classification
12   - name: TriageAndPrioritizeAlert
13     inputs: [alert_type, severity, repo_criticality]
14     actions:
15       - ComputeNormalizedPriority
16       - ApplyBusinessRules
17       - SetSLADueDates
18
19   # Assignment
20   - name: AssignToOwningTeam
21     inputs: [repo_name, alert_type, priority]
22     actions:
23       - LookupCMDBApplication
24       - ResolveOwnerFromCodeowners
25       - FallbackToTriageQueue
26       - NotifyAssignee
27
28   # Lifecycle Management
29   - name: HandleAlertStateChange
30     inputs: [current_state, new_state, reason]
31     actions:
32       - ValidateStateTransition
```

```

33     - UpdateVulnerableItem
34     - SyncToGitHub (if bidirectional)
35     - RecordAuditTrail
36
37 # Exception Handling
38 - name: ProcessExceptionRequest
39   inputs: [alert_key, reason, approver, expiration]
40   actions:
41     - ValidateApproverAuthority
42     - RecordCompensatingControls
43     - SetExpirationReminder
44     - UpdateGitHubDismissalReason

```

Listing 6: Flow Designer Subflow Architecture

8 Implementation: Webhook Receiver

8.1 Node.js/TypeScript Implementation

The following production-ready implementation includes signature validation, structured logging, queue integration, and comprehensive error handling:

```

1  import express, { Request, Response, NextFunction } from 'express';
2  import crypto from 'crypto';
3  import { SQSClient, SendMessageCommand } from '@aws-sdk/client-sqs';
4  import pino from 'pino';
5
6  // Configuration
7  const config = {
8    port: parseInt(process.env.PORT || '8080'),
9    webhookSecret: process.env.GITHUB_WEBHOOK_SECRET!,
10   sqsQueueUrl: process.env.SQS_QUEUE_URL!,
11   awsRegion: process.env.AWS_REGION || 'us-east-1',
12   servicenowUrl: process.env.SERVICENOW_INGEST_URL,
13   servicenowToken: process.env.SERVICENOW_BEARER_TOKEN,
14 };
15
16 // Initialize clients
17 const logger = pino({ level: 'info' });
18 const sqs = new SQSClient({ region: config.awsRegion });
19 const app = express();
20
21 // CRITICAL: Use raw body for HMAC verification
22 app.use('/webhook', express.raw({ type: 'application/json', limit: '5mb'
23   ' }));
24 app.use(express.json());
25
26 // Interfaces
27 interface WebhookHeaders {
28   signature: string;
29   event: string;
30   delivery: string;
31 }

```

```

31
32 interface ProcessedEvent {
33     deliveryId: string;
34     eventType: string;
35     action: string;
36     alertKey: string;
37     repository: string;
38     alertNumber: number;
39     severity: string | null;
40     state: string;
41     assignees: string[];
42     receivedAt: string;
43     payload: object;
44 }
45
46 // Timing-safe comparison to prevent timing attacks
47 function timingSafeEqual(a: string, b: string): boolean {
48     if (!a || !b) return false;
49     const bufA = Buffer.from(a);
50     const bufB = Buffer.from(b);
51     if (bufA.length !== bufB.length) return false;
52     return crypto.timingSafeEqual(bufA, bufB);
53 }
54
55 // Verify GitHub webhook signature
56 function verifySignature(payload: Buffer, signature: string): boolean {
57     const hmac = crypto.createHmac('sha256', config.webhookSecret);
58     hmac.update(payload);
59     const expected = 'sha256=${hmac.digest('hex')}';
60     return timingSafeEqual(signature, expected);
61 }
62
63 // Extract headers with validation
64 function extractHeaders(req: Request): WebhookHeaders {
65     return {
66         signature: req.header('X-Hub-Signature-256') || '',
67         event: req.header('X-GitHub-Event') || '',
68         delivery: req.header('X-GitHub-Delivery') || '',
69     };
70 }
71
72 // Compute canonical alert key
73 function computeAlertKey(event: string, payload: any): string {
74     const repo = payload.repository?.full_name || 'unknown';
75     const alertNumber = payload.alert?.number || 0;
76
77     const alertTypeMap: Record<string, string> = {
78         'code_scanning_alert': 'code_scanning',
79         'dependabot_alert': 'dependabot',
80         'secret_scanning_alert': 'secret_scanning',
81     };
82
83     const alertType = alertTypeMap[event] || event;
84     return `github:${repo}:${alertType}:${alertNumber}`;

```



```

85 }
86
87 // Extract severity from various alert types
88 function extractSeverity(event: string, payload: any): string | null {
89     switch (event) {
90         case 'code_scanning_alert':
91             return payload.alert?.rule?.severity ||
92                 payload.alert?.rule?.security_severity_level || null;
93         case 'dependabot_alert':
94             return payload.alert?.security_advisory?.severity ||
95                 payload.alert?.security_vulnerability?.severity || null;
96         case 'secret_scanning_alert':
97             // Secret scanning doesn't have severity; return 'critical' by
98             // default
99             return 'critical';
100         default:
101             return null;
102     }
103 }
104
105 // Extract assignees (new 2025 feature)
106 function extractAssignees(payload: any): string[] {
107     const assignees = payload.alert?.assignees || [];
108     return assignees.map((a: any) => a.login);
109 }
110
111 // Process and normalize the webhook event
112 function processEvent(headers: WebhookHeaders, payload: any):
113     ProcessedEvent {
114     return {
115         deliveryId: headers.delivery,
116         eventType: headers.event,
117         action: payload.action || 'unknown',
118         alertKey: computeAlertKey(headers.event, payload),
119         repository: payload.repository?.full_name || 'unknown',
120         alertNumber: payload.alert?.number || 0,
121         severity: extractSeverity(headers.event, payload),
122         state: payload.alert?.state || 'unknown',
123         assignees: extractAssignees(payload),
124         receivedAt: new Date().toISOString(),
125         payload: sanitizePayload(headers.event, payload),
126     };
127 }
128
129 // Sanitize payload to remove sensitive data
130 function sanitizePayload(event: string, payload: any): object {
131     if (event === 'secret_scanning_alert') {
132         // CRITICAL: Never store actual secret values
133         const sanitized = { ...payload };
134         if (sanitized.alert) {
135             sanitized.alert = {
136                 ...sanitized.alert,
137                 secret: '[REDACTED]',
138             };
139         }
140     }
141 }

```

```
137     }
138     return sanitized;
139 }
140 return payload;
141 }
142
143 // Publish to SQS queue
144 async function publishToQueue(event: ProcessedEvent): Promise<void> {
145     const command = new SendMessageCommand({
146         QueueUrl: config.sqsQueueUrl,
147         MessageBody: JSON.stringify(event),
148         MessageAttributes: {
149             EventType: { DataType: 'String', StringValue: event.eventType },
150             AlertKey: { DataType: 'String', StringValue: event.alertKey },
151             Priority: { DataType: 'String', StringValue: event.severity || '
medium' },
152         },
153         MessageDeduplicationId: event.deliveryId,
154         MessageGroupId: event.repository.replace('/', '-'),
155     });
156
157     await sqs.send(command);
158 }
159
160 // Allowed event types
161 const ALLOWED_EVENTS = new Set([
162     'code_scanning_alert',
163     'dependabot_alert',
164     'secret_scanning_alert',
165     'security_and_analysis',
166 ]);
167
168 // Main webhook handler
169 app.post('/webhook/github', async (req: Request, res: Response) => {
170     const startTime = Date.now();
171     const headers = extractHeaders(req);
172
173     // Log incoming request
174     logger.info({
175         delivery: headers.delivery,
176         event: headers.event,
177         userAgent: req.header('User-Agent'),
178     }, 'Webhook received');
179
180     // Validate signature
181     if (!verifySignature(req.body as Buffer, headers.signature)) {
182         logger.warn({ delivery: headers.delivery }, 'Invalid signature');
183         return res.status(401).json({
184             error: 'invalid_signature',
185             message: 'Webhook signature validation failed',
186         });
187     }
188
189     // Validate event type
```

```
190 if (!ALLOWED_EVENTS.has(headers.event)) {
191   logger.info({
192     delivery: headers.delivery,
193     event: headers.event,
194   }, 'Ignored event type');
195   return res.status(200).json({
196     status: 'ignored',
197     reason: 'event_type_not_processed',
198   });
199 }
200
201 try {
202   const payload = JSON.parse(req.body.toString());
203   const event = processEvent(headers, payload);
204
205   // Publish to queue for async processing
206   await publishToQueue(event);
207
208   const duration = Date.now() - startTime;
209   logger.info({
210     delivery: headers.delivery,
211     alertKey: event.alertKey,
212     action: event.action,
213     duration,
214   }, 'Event queued successfully');
215
216   return res.status(202).json({
217     status: 'accepted',
218     delivery_id: headers.delivery,
219     alert_key: event.alertKey,
220   });
221 } catch (error) {
222   logger.error({
223     delivery: headers.delivery,
224     error: error instanceof Error ? error.message : 'Unknown error',
225   }, 'Processing failed');
226
227   // Return 500 so GitHub will retry
228   return res.status(500).json({
229     error: 'processing_failed',
230     delivery_id: headers.delivery,
231   });
232 }
233 }
234 });
235
236 // Health check endpoint
237 app.get('/health', (req: Request, res: Response) => {
238   res.status(200).json({
239     status: 'healthy',
240     timestamp: new Date().toISOString(),
241   });
242 });
243
```

```

244 // Start server
245 app.listen(config.port, () => {
246     logger.info({ port: config.port }, 'Webhook receiver started');
247 });

```

Listing 7: TypeScript Webhook Receiver with Full Validation

8.2 Go Implementation

High-performance Go implementation for high-volume deployments:

```

1  package main
2
3  import (
4      "context"
5      "crypto/hmac"
6      "crypto/sha256"
7      "crypto/subtle"
8      "encoding/hex"
9      "encoding/json"
10     "fmt"
11     "io"
12     "log/slog"
13     "net/http"
14     "os"
15     "strings"
16     "time"
17
18     "github.com/aws/aws-sdk-go-v2/config"
19     "github.com/aws/aws-sdk-go-v2/service/sqs"
20 )
21
22 type Config struct {
23     Port          string
24     WebhookSecret string
25     SQSQueueURL   string
26 }
27
28 type WebhookEvent struct {
29     DeliveryID   string    `json:"delivery_id"`
30     EventType    string    `json:"event_type"`
31     Action       string    `json:"action"`
32     AlertKey     string    `json:"alert_key"`
33     Repository   string    `json:"repository"`
34     AlertNumber  int       `json:"alert_number"`
35     Severity     string    `json:"severity,omitempty"`
36     State        string    `json:"state"`
37     Assignees    []string  `json:"assignees"`
38     ReceivedAt   time.Time `json:"received_at"`
39     Payload      any       `json:"payload"`
40 }
41
42 type WebhookHandler struct {
43     config *Config

```

```

44     sqsClient *sqs.Client
45     logger    *slog.Logger
46 }
47
48 func NewWebhookHandler(cfg *Config) (*WebhookHandler, error) {
49     awsCfg, err := config.LoadDefaultConfig(context.Background())
50     if err != nil {
51         return nil, fmt.Errorf("failed to load AWS config: %w", err)
52     }
53
54     return &WebhookHandler{
55         config:    cfg,
56         sqsClient: sqs.NewFromConfig(awsCfg),
57         logger:    slog.New(slog.NewJSONHandler(os.Stdout, nil)),
58     }, nil
59 }
60
61 func (h *WebhookHandler) verifySignature(payload []byte, signature
    string) bool {
62     if !strings.HasPrefix(signature, "sha256=") {
63         return false
64     }
65
66     mac := hmac.New(sha256.New, []byte(h.config.WebhookSecret))
67     mac.Write(payload)
68     expected := "sha256=" + hex.EncodeToString(mac.Sum(nil))
69
70     return subtle.ConstantTimeCompare([]byte(signature), []byte(expected)
71     ) == 1
72 }
73
74 func (h *WebhookHandler) computeAlertKey(eventType string, payload map[
75     string]any) string {
76     repo := "unknown"
77     if r, ok := payload["repository"].(map[string]any); ok {
78         if fn, ok := r["full_name"].(string); ok {
79             repo = fn
80         }
81     }
82
83     var alertNumber float64
84     if alert, ok := payload["alert"].(map[string]any); ok {
85         if num, ok := alert["number"].(float64); ok {
86             alertNumber = num
87         }
88     }
89
90     alertTypeMap := map[string]string{
91         "code_scanning_alert": "code_scanning",
92         "dependabot_alert":    "dependabot",
93         "secret_scanning_alert": "secret_scanning",
94     }
95
96     alertType := alertTypeMap[eventType]

```

```

95     if alertType == "" {
96         alertType = eventType
97     }
98
99     return fmt.Sprintf("github:%s:%s:%.0f", repo, alertType, alertNumber)
100 }
101
102 func (h *WebhookHandler) extractSeverity(eventType string, payload map[
103     string]any) string {
104     alert, ok := payload["alert"].(map[string]any)
105     if !ok {
106         return ""
107     }
108     switch eventType {
109     case "code_scanning_alert":
110         if rule, ok := alert["rule"].(map[string]any); ok {
111             if sev, ok := rule["severity"].(string); ok {
112                 return sev
113             }
114             if sev, ok := rule["security_severity_level"].(string); ok {
115                 return sev
116             }
117         }
118     case "dependabot_alert":
119         if adv, ok := alert["security_advisory"].(map[string]any); ok {
120             if sev, ok := adv["severity"].(string); ok {
121                 return sev
122             }
123         }
124     case "secret_scanning_alert":
125         return "critical"
126     }
127     return ""
128 }
129
130 func (h *WebhookHandler) extractAssignees(payload map[string]any) []
131     string {
132     var assignees []string
133     alert, ok := payload["alert"].(map[string]any)
134     if !ok {
135         return assignees
136     }
137     if ass, ok := alert["assignees"].([]any); ok {
138         for _, a := range ass {
139             if user, ok := a.(map[string]any); ok {
140                 if login, ok := user["login"].(string); ok {
141                     assignees = append(assignees, login)
142                 }
143             }
144         }
145     }
146     return assignees

```

```

147 }
148
149 func (h *WebhookHandler) processEvent(
150     deliveryID, eventType string,
151     payload map[string]any,
152 ) *WebhookEvent {
153     action := ""
154     if a, ok := payload["action"].(string); ok {
155         action = a
156     }
157
158     state := "unknown"
159     if alert, ok := payload["alert"].(map[string]any); ok {
160         if s, ok := alert["state"].(string); ok {
161             state = s
162         }
163     }
164
165     var alertNumber int
166     if alert, ok := payload["alert"].(map[string]any); ok {
167         if num, ok := alert["number"].(float64); ok {
168             alertNumber = int(num)
169         }
170     }
171
172     repo := "unknown"
173     if r, ok := payload["repository"].(map[string]any); ok {
174         if fn, ok := r["full_name"].(string); ok {
175             repo = fn
176         }
177     }
178
179     // Sanitize secret scanning payloads
180     sanitizedPayload := payload
181     if eventType == "secret_scanning_alert" {
182         if alert, ok := sanitizedPayload["alert"].(map[string]any); ok {
183             alert["secret"] = "[REDACTED]"
184         }
185     }
186
187     return &WebhookEvent{
188         DeliveryID: deliveryID,
189         EventType:  eventType,
190         Action:     action,
191         AlertKey:   h.computeAlertKey(eventType, payload),
192         Repository: repo,
193         AlertNumber: alertNumber,
194         Severity:   h.extractSeverity(eventType, payload),
195         State:      state,
196         Assignees:  h.extractAssignees(payload),
197         ReceivedAt: time.Now().UTC(),
198         Payload:    sanitizedPayload,
199     }
200 }

```

```

201
202 func (h *WebhookHandler) publishToQueue(ctx context.Context, event *
    WebhookEvent) error {
203     body, err := json.Marshal(event)
204     if err != nil {
205         return fmt.Errorf("failed to marshal event: %w", err)
206     }
207
208     messageGroupID := strings.ReplaceAll(event.Repository, "/", "-")
209
210     _, err = h.sqsClient.SendMessage(ctx, &sqs.SendMessageInput{
211         QueueUrl:           &h.config.SQSQueueURL,
212         MessageBody:         ptrString(string(body)),
213         MessageDeduplicationId: &event.DeliveryID,
214         MessageGroupId:       &messageGroupID,
215     })
216
217     return err
218 }
219
220 func ptrString(s string) *string { return &s }
221
222 var allowedEvents = map[string]bool{
223     "code_scanning_alert": true,
224     "dependabot_alert":    true,
225     "secret_scanning_alert": true,
226     "security_and_analysis": true,
227 }
228
229 func (h *WebhookHandler) ServeHTTP(w http.ResponseWriter, r *http.
    Request) {
230     if r.Method != http.MethodPost {
231         http.Error(w, "Method not allowed", http.StatusMethodNotAllowed)
232         return
233     }
234
235     // Extract headers
236     signature := r.Header.Get("X-Hub-Signature-256")
237     eventType := r.Header.Get("X-GitHub-Event")
238     deliveryID := r.Header.Get("X-GitHub-Delivery")
239
240     // Read body
241     body, err := io.ReadAll(io.LimitReader(r.Body, 5<<20)) // 5MB limit
242     if err != nil {
243         h.logger.Error("Failed to read body", "delivery", deliveryID, "
            error", err)
244         http.Error(w, "Failed to read body", http.StatusBadRequest)
245         return
246     }
247
248     // Verify signature
249     if !h.verifySignature(body, signature) {
250         h.logger.Warn("Invalid signature", "delivery", deliveryID)
251         w.WriteHeader(http.StatusUnauthorized)

```



```
252     json.NewEncoder(w).Encode(map[string]string{
253         "error":    "invalid_signature",
254         "message": "Webhook signature validation failed",
255     })
256     return
257 }
258
259 // Check event type
260 if !allowedEvents[eventType] {
261     h.logger.Info("Ignored event type", "delivery", deliveryID, "event "
262         , eventType)
263     w.WriteHeader(http.StatusOK)
264     json.NewEncoder(w).Encode(map[string]string{
265         "status": "ignored",
266         "reason": "event_type_not_processed",
267     })
268     return
269 }
270
271 // Parse payload
272 var payload map[string]any
273 if err := json.Unmarshal(body, &payload); err != nil {
274     h.logger.Error("Failed to parse payload", "delivery", deliveryID, "
275         error", err)
276     http.Error(w, "Invalid JSON payload", http.StatusBadRequest)
277     return
278 }
279
280 // Process and queue
281 event := h.processEvent(deliveryID, eventType, payload)
282
283 ctx, cancel := context.WithTimeout(r.Context(), 5*time.Second)
284 defer cancel()
285
286 if err := h.publishToQueue(ctx, event); err != nil {
287     h.logger.Error("Failed to queue event",
288         "delivery", deliveryID,
289         "alert_key", event.AlertKey,
290         "error", err,
291     )
292     http.Error(w, "Failed to queue event", http.
293         StatusInternalServerError)
294     return
295 }
296
297 h.logger.Info("Event queued",
298     "delivery", deliveryID,
299     "alert_key", event.AlertKey,
300     "action", event.Action,
301 )
302
303 w.WriteHeader(http.StatusAccepted)
304 json.NewEncoder(w).Encode(map[string]string{
305     "status":    "accepted",
```

```

303     "delivery_id": deliveryID,
304     "alert_key":   event.AlertKey,
305   })
306 }
307
308 func main() {
309     cfg := &Config{
310         Port:           getEnv("PORT", "8080"),
311         WebhookSecret:  os.Getenv("GITHUB_WEBHOOK_SECRET"),
312         SQSQueueURL:    os.Getenv("SQS_QUEUE_URL"),
313     }
314
315     handler, err := NewWebhookHandler(cfg)
316     if err != nil {
317         slog.Error("Failed to create handler", "error", err)
318         os.Exit(1)
319     }
320
321     mux := http.NewServeMux()
322     mux.Handle("/webhook/github", handler)
323     mux.HandleFunc("/health", func(w http.ResponseWriter, r *http.Request) {
324         json.NewEncoder(w).Encode(map[string]string{
325             "status":    "healthy",
326             "timestamp": time.Now().UTC().Format(time.RFC3339),
327         })
328     })
329
330     slog.Info("Starting server", "port", cfg.Port)
331     if err := http.ListenAndServe(":"+cfg.Port, mux); err != nil {
332         slog.Error("Server failed", "error", err)
333         os.Exit(1)
334     }
335 }
336
337 func getEnv(key, fallback string) string {
338     if v := os.Getenv(key); v != "" {
339         return v
340     }
341     return fallback
342 }

```

Listing 8: Go Webhook Receiver with Concurrent Processing

8.3 ServiceNow Scripted REST API Implementation

For deployments preferring a ServiceNow-native approach:

```

1  // Scripted REST API Resource: POST /api/x_ghas/webhook
2  // Table: x_ghas_event (custom staging table)
3
4  (function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/
5      response) {
6      'use strict';

```

```
6
7     var startTime = new GlideDateTime();
8
9     // Configuration
10    var SECRET = gs.getProperty('x_ghas.webhook.secret');
11    var MAX_PAYLOAD_SIZE = 5242880; // 5MB
12
13    // Extract headers
14    var signature = request.getHeader('X-Hub-Signature-256') || '';
15    var eventType = request.getHeader('X-GitHub-Event') || '';
16    var deliveryId = request.getHeader('X-GitHub-Delivery') || '';
17
18    // Get raw body for signature verification
19    var bodyStr = request.body.dataString || '';
20
21    // Check payload size
22    if (bodyStr.length > MAX_PAYLOAD_SIZE) {
23        response.setStatus(413);
24        response.setBody({
25            error: 'payload_too_large',
26            max_size: MAX_PAYLOAD_SIZE
27        });
28        return;
29    }
30
31    // Verify signature using Java crypto
32    var expectedSignature = 'sha256=' + computeHmacSha256(SECRET,
bodyStr);
33    if (!constantTimeEquals(signature, expectedSignature)) {
34        gs.warn('GHAS Webhook: Invalid signature for delivery ' +
deliveryId);
35        response.setStatus(401);
36        response.setBody({
37            error: 'invalid_signature',
38            delivery_id: deliveryId
39        });
40        return;
41    }
42
43    // Validate event type
44    var allowedEvents = [
45        'code_scanning_alert',
46        'dependabot_alert',
47        'secret_scanning_alert',
48        'security_and_analysis'
49    ];
50
51    if (allowedEvents.indexOf(eventType) === -1) {
52        response.setStatus(200);
53        response.setBody({
54            status: 'ignored',
55            reason: 'event_type_not_processed',
56            event_type: eventType
57        });
```

```
58     return;
59 }
60
61 try {
62     var payload = JSON.parse(bodyStr);
63
64     // Compute canonical alert key
65     var alertKey = computeAlertKey(eventType, payload);
66
67     // Check for duplicate delivery (idempotency)
68     var existingEvent = new GlideRecord('x_ghas_event');
69     existingEvent.addQuery('u_delivery_id', deliveryId);
70     existingEvent.query();
71
72     if (existingEvent.next()) {
73         response.setStatus(200);
74         response.setBody({
75             status: 'duplicate',
76             delivery_id: deliveryId,
77             existing_sys_id: existingEvent.sys_id.toString()
78         });
79         return;
80     }
81
82     // Create staging record
83     var eventRecord = new GlideRecord('x_ghas_event');
84     eventRecord.initialize();
85     eventRecord.u_delivery_id = deliveryId;
86     eventRecord.u_event_type = eventType;
87     eventRecord.u_action = payload.action || 'unknown';
88     eventRecord.u_alert_key = alertKey;
89     eventRecord.u_repository = getNestedValue(payload, 'repository.
full_name', 'unknown');
90     eventRecord.u_alert_number = getNestedValue(payload, 'alert.
number', 0);
91     eventRecord.u_severity = extractSeverity(eventType, payload);
92     eventRecord.u_state = getNestedValue(payload, 'alert.state', '
unknown');
93     eventRecord.u_assignees = JSON.stringify(extractAssignees(
payload));
94     eventRecord.u_processing_state = 'received';
95     eventRecord.u_received_at = startTime;
96
97     // Sanitize and store payload
98     var sanitizedPayload = sanitizePayload(eventType, payload);
99     eventRecord.u_payload = JSON.stringify(sanitizedPayload);
100
101     var sysId = eventRecord.insert();
102
103     if (!sysId) {
104         throw new Error('Failed to insert event record');
105     }
106
107     // Queue for async processing via event
```

```

108     gs.eventQueue('x_ghas.event.received', eventRecord, deliveryId,
109         eventType);
110
111     // Calculate processing time
112     var endTime = new GlideDateTime();
113     var duration = GlideDateTime.subtract(startTime, endTime).
114         getNumericValue();
115
116     gs.info('GHAS Webhook: Queued event ' + deliveryId +
117         ' (' + alertKey + ') in ' + duration + 'ms');
118
119     response.setStatus(202);
120     response.setBody({
121         status: 'accepted',
122         delivery_id: deliveryId,
123         alert_key: alertKey,
124         sys_id: sysId,
125         processing_time_ms: duration
126     });
127
128 } catch (e) {
129     gs.error('GHAS Webhook: Processing error for ' + deliveryId + '
130 : ' + e.message);
131     response.setStatus(500);
132     response.setBody({
133         error: 'processing_failed',
134         delivery_id: deliveryId,
135         message: e.message
136     });
137 }
138
139 // ===== Helper Functions =====
140
141 function computeHmacSha256(key, data) {
142     var Mac = Packages.java.lang.Security.Mac;
143     var SecretKeySpec = Packages.java.lang.spec.SecretKeySpec;
144     var StandardCharsets = Packages.java.nio.charset.
145 StandardCharsets;
146
147     var mac = Mac.getInstance('HmacSHA256');
148     var keyBytes = new java.lang.String(key).getBytes(
149 StandardCharsets.UTF_8);
150     var keySpec = new SecretKeySpec(keyBytes, 'HmacSHA256');
151     mac.init(keySpec);
152
153     var dataBytes = new java.lang.String(data).getBytes(
154 StandardCharsets.UTF_8);
155     var hashBytes = mac.doFinal(dataBytes);
156
157     var sb = new java.lang.StringBuilder();
158     for (var i = 0; i < hashBytes.length; i++) {
159         sb.append(java.lang.String.format('%02x', hashBytes[i] & 0
160 xff));
161     }

```

```
155     return sb.toString();
156 }
157
158 function constantTimeEquals(a, b) {
159     if (!a || !b || a.length !== b.length) return false;
160     var result = 0;
161     for (var i = 0; i < a.length; i++) {
162         result |= a.charCodeAt(i) ^ b.charCodeAt(i);
163     }
164     return result === 0;
165 }
166
167 function computeAlertKey(eventType, payload) {
168     var repo = getNestedValue(payload, 'repository.full_name', '
unknown');
169     var alertNumber = getNestedValue(payload, 'alert.number', 0);
170
171     var typeMap = {
172         'code_scanning_alert': 'code_scanning',
173         'dependabot_alert': 'dependabot',
174         'secret_scanning_alert': 'secret_scanning'
175     };
176
177     var alertType = typeMap[eventType] || eventType;
178     return 'github:' + repo + ':' + alertType + ':' + alertNumber;
179 }
180
181 function extractSeverity(eventType, payload) {
182     var alert = payload.alert || {};
183
184     switch (eventType) {
185         case 'code_scanning_alert':
186             var rule = alert.rule || {};
187             return rule.severity || rule.security_severity_level ||
'';
188         case 'dependabot_alert':
189             var advisory = alert.security_advisory || {};
190             return advisory.severity || '';
191         case 'secret_scanning_alert':
192             return 'critical';
193         default:
194             return '';
195     }
196 }
197
198 function extractAssignees(payload) {
199     var assignees = [];
200     var alertAssignees = getNestedValue(payload, 'alert.assignees',
[]);
201
202     for (var i = 0; i < alertAssignees.length; i++) {
203         if (alertAssignees[i] && alertAssignees[i].login) {
204             assignees.push(alertAssignees[i].login);
205         }
206     }
207 }
```

```

206     }
207     return assignees;
208 }
209
210 function sanitizePayload(eventType, payload) {
211     if (eventType === 'secret_scanning_alert') {
212         var sanitized = JSON.parse(JSON.stringify(payload));
213         if (sanitized.alert) {
214             sanitized.alert.secret = '[REDACTED]';
215         }
216         return sanitized;
217     }
218     return payload;
219 }
220
221 function getNestedValue(obj, path, defaultValue) {
222     var parts = path.split('.');
223     var current = obj;
224
225     for (var i = 0; i < parts.length; i++) {
226         if (current === null || current === undefined) {
227             return defaultValue;
228         }
229         current = current[parts[i]];
230     }
231
232     return (current !== null && current !== undefined) ? current :
    defaultValue;
233 }
234
235 })(request, response);

```

Listing 9: ServiceNow Scripted REST API Resource

9 Implementation: Reconciliation Worker

9.1 Python Reconciliation Service

Production-grade Python implementation with comprehensive API support:

```

1  #!/usr/bin/env python3
2  """
3  GitHub Security Alerts Reconciliation Worker
4
5  Synchronizes GitHub security alerts with ServiceNow, supporting:
6  - Code scanning, secret scanning, and Dependabot alerts
7  - Alert assignees (new 2025 feature)
8  - Security campaigns
9  - Bidirectional state synchronization
10 """
11
12 import os
13 import sys

```

```

14 import json
15 import logging
16 import hashlib
17 from datetime import datetime, timedelta, timezone
18 from typing import Any, Iterator, Optional
19 from dataclasses import dataclass, asdict
20 from enum import Enum
21
22 import requests
23 from requests.adapters import HTTPAdapter
24 from urllib3.util.retry import Retry
25
26 # Configure logging
27 logging.basicConfig(
28     level=logging.INFO,
29     format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
30 )
31 logger = logging.getLogger(__name__)
32
33
34 class AlertType(Enum):
35     CODE_SCANNING = "code_scanning"
36     SECRET_SCANNING = "secret_scanning"
37     DEPENDABOT = "dependabot"
38
39
40 @dataclass
41 class GitHubConfig:
42     """GitHub API configuration."""
43     token: str
44     api_base: str = "https://api.github.com"
45     api_version: str = "2022-11-28"
46
47     @property
48     def headers(self) -> dict[str, str]:
49         return {
50             "Accept": "application/vnd.github+json",
51             "Authorization": f"Bearer {self.token}",
52             "X-GitHub-API-Version": self.api_version,
53         }
54
55
56 @dataclass
57 class ServiceNowConfig:
58     """ServiceNow API configuration."""
59     instance_url: str
60     username: str
61     password: str
62     ingest_endpoint: str = "/api/x_ghas/reconciliation"
63
64     @property
65     def full_url(self) -> str:
66         return f"{self.instance_url.rstrip('/')}{self.ingest_endpoint}"
67

```



```
68
69 @dataclass
70 class NormalizedAlert:
71     """Normalized alert structure for ServiceNow ingestion."""
72     provider: str
73     alert_type: str
74     organization: str
75     repository: str
76     alert_number: int
77     canonical_key: str
78     state: str
79     severity: Optional[str]
80     created_at: str
81     updated_at: str
82     html_url: str
83     rule_id: Optional[str]
84     rule_description: Optional[str]
85     cwe_ids: list[str]
86     assignees: list[str]
87     dismissed_by: Optional[str]
88     dismissed_reason: Optional[str]
89     fixed_at: Optional[str]
90     raw_alert: dict[str, Any]
91
92
93 class GitHubClient:
94     """GitHub API client with retry logic and pagination support."""
95
96     def __init__(self, config: GitHubConfig):
97         self.config = config
98         self.session = self._create_session()
99
100     def _create_session(self) -> requests.Session:
101         session = requests.Session()
102
103         # Configure retries with exponential backoff
104         retry_strategy = Retry(
105             total=5,
106             backoff_factor=1,
107             status_forcelist=[429, 500, 502, 503, 504],
108             allowed_methods=["GET", "PATCH"],
109             respect_retry_after_header=True,
110         )
111
112         adapter = HTTPAdapter(max_retries=retry_strategy)
113         session.mount("https://", adapter)
114         session.headers.update(self.config.headers)
115
116         return session
117
118     def _paginate(
119         self,
120         url: str,
121         params: Optional[dict[str, Any]] = None
```

```

122     ) -> Iterator[dict[str, Any]]:
123         """Paginate through GitHub API results."""
124         params = params or {}
125         params.setdefault("per_page", 100)
126
127         while url:
128             response = self.session.get(url, params=params, timeout=30)
129             response.raise_for_status()
130
131             # Check rate limits
132             remaining = int(response.headers.get("X-RateLimit-Remaining", 0))
133             if remaining < 100:
134                 reset_time = int(response.headers.get("X-RateLimit-Reset", 0))
135                 logger.warning(
136                     f"Rate limit low: {remaining} remaining, "
137                     f"resets at {datetime.fromtimestamp(reset_time)}"
138                 )
139
140                 data = response.json()
141                 if isinstance(data, list):
142                     yield from data
143                 else:
144                     yield data
145
146             # Get next page URL from Link header
147             url = None
148             params = None # Only apply params on first request
149
150             link_header = response.headers.get("Link", "")
151             for link in link_header.split(","):
152                 if 'rel="next"' in link:
153                     url = link[link.find("<") + 1:link.find(">")]
154                     break
155
156     def list_code_scanning_alerts(
157         self,
158         owner: str,
159         repo: str,
160         state: str = "open",
161         since: Optional[datetime] = None,
162     ) -> Iterator[dict[str, Any]]:
163         """List code scanning alerts for a repository."""
164         url = f"{self.config.api_base}/repos/{owner}/{repo}/code-scanning/alerts"
165         params = {"state": state}
166
167         if since:
168             # Filter by tool_name and created_at if needed
169             pass
170
171         yield from self._paginate(url, params)
172

```

```

173     def list_secret_scanning_alerts(
174         self,
175         owner: str,
176         repo: str,
177         state: str = "open",
178     ) -> Iterator[dict[str, Any]]:
179         """List secret scanning alerts for a repository."""
180         url = f"{self.config.api_base}/repos/{owner}/{repo}/secret-
scanning/alerts"
181         params = {"state": state}
182         yield from self._paginate(url, params)
183
184     def list_dependabot_alerts(
185         self,
186         owner: str,
187         repo: str,
188         state: str = "open",
189     ) -> Iterator[dict[str, Any]]:
190         """List Dependabot alerts for a repository."""
191         url = f"{self.config.api_base}/repos/{owner}/{repo}/dependabot/
alerts"
192         params = {"state": state}
193         yield from self._paginate(url, params)
194
195     def list_org_repos(
196         self,
197         org: str,
198         repo_type: str = "all",
199     ) -> Iterator[dict[str, Any]]:
200         """List repositories in an organization."""
201         url = f"{self.config.api_base}/orgs/{org}/repos"
202         params = {"type": repo_type, "sort": "updated", "direction": "
desc"}
203         yield from self._paginate(url, params)
204
205     def update_alert_assignees(
206         self,
207         owner: str,
208         repo: str,
209         alert_type: AlertType,
210         alert_number: int,
211         assignees: list[str],
212     ) -> dict[str, Any]:
213         """Update alert assignees (bidirectional sync)."""
214         if alert_type == AlertType.CODE_SCANNING:
215             url = f"{self.config.api_base}/repos/{owner}/{repo}/code-
scanning/alerts/{alert_number}"
216         elif alert_type == AlertType.SECRET_SCANNING:
217             url = f"{self.config.api_base}/repos/{owner}/{repo}/secret-
scanning/alerts/{alert_number}"
218         else:
219             raise ValueError(f"Assignees not supported for {alert_type}
")
220

```

```

221         response = self.session.patch(
222             url,
223             json={"assignees": assignees},
224             timeout=30,
225         )
226         response.raise_for_status()
227         return response.json()
228
229
230 class AlertNormalizer:
231     """Normalize GitHub alerts to common format."""
232
233     @staticmethod
234     def compute_canonical_key(
235         org: str,
236         repo: str,
237         alert_type: AlertType,
238         alert_number: int,
239     ) -> str:
240         """Compute deterministic canonical key."""
241         return f"github:{org}/{repo}:{alert_type.value}:{alert_number}"
242
243     @classmethod
244     def normalize_code_scanning(
245         cls,
246         org: str,
247         repo: str,
248         alert: dict[str, Any],
249     ) -> NormalizedAlert:
250         """Normalize code scanning alert."""
251         rule = alert.get("rule", {})
252
253         # Extract CWE IDs from tags
254         cwe_ids = [
255             tag.replace("external/cwe/", "")
256             for tag in rule.get("tags", [])
257             if tag.startswith("external/cwe/")
258         ]
259
260         # Extract assignees
261         assignees = [a.get("login", "") for a in alert.get("assignees",
262             [])]
263
264         return NormalizedAlert(
265             provider="github",
266             alert_type=AlertType.CODE_SCANNING.value,
267             organization=org,
268             repository=repo,
269             alert_number=alert.get("number", 0),
270             canonical_key=cls.compute_canonical_key(
271                 org, repo, AlertType.CODE_SCANNING, alert.get("number",
272                 0)
273             ),
274             state=alert.get("state", "unknown"),

```

```

273         severity=rule.get("severity") or rule.get("
security_severity_level"),
274         created_at=alert.get("created_at", ""),
275         updated_at=alert.get("updated_at", ""),
276         html_url=alert.get("html_url", ""),
277         rule_id=rule.get("id"),
278         rule_description=rule.get("description"),
279         cwe_ids=cwe_ids,
280         assignees=assignees,
281         dismissed_by=alert.get("dismissed_by", {}).get("login") if
alert.get("dismissed_by") else None,
282         dismissed_reason=alert.get("dismissed_reason"),
283         fixed_at=alert.get("fixed_at"),
284         raw_alert=alert,
285     )
286
287     @classmethod
288     def normalize_secret_scanning(
289         cls,
290         org: str,
291         repo: str,
292         alert: dict[str, Any],
293     ) -> NormalizedAlert:
294         """Normalize secret scanning alert.
295
296         CRITICAL: Never store actual secret values.
297         """
298         # Redact secret value
299         sanitized_alert = {**alert}
300         if "secret" in sanitized_alert:
301             sanitized_alert["secret"] = "[REDACTED]"
302
303         assignees = [a.get("login", "") for a in alert.get("assignees",
[])]
304
305         return NormalizedAlert(
306             provider="github",
307             alert_type=AlertType.SECRET_SCANNING.value,
308             organization=org,
309             repository=repo,
310             alert_number=alert.get("number", 0),
311             canonical_key=cls.compute_canonical_key(
312                 org, repo, AlertType.SECRET_SCANNING, alert.get("number
", 0)
313             ),
314             state=alert.get("state", "unknown"),
315             severity="critical", # Secret exposure is always critical
316             created_at=alert.get("created_at", ""),
317             updated_at=alert.get("updated_at", ""),
318             html_url=alert.get("html_url", ""),
319             rule_id=alert.get("secret_type"),
320             rule_description=f"Exposed {alert.get('
secret_type_display_name', 'secret')}",
321             cwe_ids=["CWE-798"], # Hard-coded credentials

```

```

322         assignees=assignees,
323         dismissed_by=alert.get("resolved_by", {}).get("login") if
alert.get("resolved_by") else None,
324         dismissed_reason=alert.get("resolution"),
325         fixed_at=alert.get("resolved_at"),
326         raw_alert=sanitized_alert,
327     )
328
329     @classmethod
330     def normalize_dependabot(
331         cls,
332         org: str,
333         repo: str,
334         alert: dict[str, Any],
335     ) -> NormalizedAlert:
336         """Normalize Dependabot alert."""
337         advisory = alert.get("security_advisory", {})
338         vulnerability = alert.get("security_vulnerability", {})
339
340         # Extract CWE IDs
341         cwe_ids = [
342             cwe.get("cwe_id", "")
343             for cwe in advisory.get("cwes", [])
344         ]
345
346         return NormalizedAlert(
347             provider="github",
348             alert_type=AlertType.DEPENDABOT.value,
349             organization=org,
350             repository=repo,
351             alert_number=alert.get("number", 0),
352             canonical_key=cls.compute_canonical_key(
353                 org, repo, AlertType.DEPENDABOT, alert.get("number", 0)
354             ),
355             state=alert.get("state", "unknown"),
356             severity=advisory.get("severity") or vulnerability.get("
severity"),
357             created_at=alert.get("created_at", ""),
358             updated_at=alert.get("updated_at", ""),
359             html_url=alert.get("html_url", ""),
360             rule_id=advisory.get("ghsa_id") or advisory.get("cve_id"),
361             rule_description=advisory.get("summary"),
362             cwe_ids=cwe_ids,
363             assignees=[], # Dependabot doesn't support assignees
364             dismissed_by=alert.get("dismissed_by", {}).get("login") if
alert.get("dismissed_by") else None,
365             dismissed_reason=alert.get("dismissed_reason"),
366             fixed_at=alert.get("fixed_at"),
367             raw_alert=alert,
368         )
369
370
371 class ServiceNowClient:
372     """ServiceNow API client."""

```

```

373
374     def __init__(self, config: ServiceNowConfig):
375         self.config = config
376         self.session = self._create_session()
377
378     def _create_session(self) -> requests.Session:
379         session = requests.Session()
380         session.auth = (self.config.username, self.config.password)
381         session.headers.update({
382             "Content-Type": "application/json",
383             "Accept": "application/json",
384         })
385         return session
386
387     def upsert_alert(self, alert: NormalizedAlert) -> dict[str, Any]:
388         """Upsert alert to ServiceNow."""
389         payload = asdict(alert)
390
391         # Don't send full raw alert to reduce payload size
392         payload["raw_alert_hash"] = hashlib.sha256(
393             json.dumps(alert.raw_alert, sort_keys=True).encode()
394         ).hexdigest()
395         del payload["raw_alert"]
396
397         response = self.session.post(
398             self.config.full_url,
399             json=payload,
400             timeout=30,
401         )
402         response.raise_for_status()
403         return response.json()
404
405     def upsert_alerts_batch(
406         self,
407         alerts: list[NormalizedAlert],
408         batch_size: int = 100,
409     ) -> dict[str, Any]:
410         """Batch upsert alerts."""
411         results = {"success": 0, "failed": 0, "errors": []}
412
413         for i in range(0, len(alerts), batch_size):
414             batch = alerts[i:i + batch_size]
415
416             payloads = []
417             for alert in batch:
418                 payload = asdict(alert)
419                 payload["raw_alert_hash"] = hashlib.sha256(
420                     json.dumps(alert.raw_alert, sort_keys=True).encode
421                 ).hexdigest()
422                 del payload["raw_alert"]
423                 payloads.append(payload)
424
425             try:

```

```

426         response = self.session.post(
427             f"{self.config.full_url}/batch",
428             json={"alerts": payloads},
429             timeout=60,
430         )
431         response.raise_for_status()
432         result = response.json()
433         results["success"] += result.get("success", 0)
434         results["failed"] += result.get("failed", 0)
435     except Exception as e:
436         logger.error(f"Batch upsert failed: {e}")
437         results["failed"] += len(batch)
438         results["errors"].append(str(e))
439
440     return results
441
442
443 class ReconciliationWorker:
444     """Main reconciliation orchestrator."""
445
446     def __init__(
447         self,
448         github_client: GitHubClient,
449         servicenow_client: ServiceNowClient,
450     ):
451         self.github = github_client
452         self.servicenow = servicenow_client
453         self.normalizer = AlertNormalizer()
454
455     def reconcile_repository(
456         self,
457         org: str,
458         repo: str,
459         alert_types: Optional[list[AlertType]] = None,
460         states: Optional[list[str]] = None,
461     ) -> dict[str, Any]:
462         """Reconcile alerts for a single repository."""
463         alert_types = alert_types or list(AlertType)
464         states = states or ["open", "dismissed", "fixed"]
465
466         results = {
467             "repository": f"{org}/{repo}",
468             "timestamp": datetime.now(timezone.utc).isoformat(),
469             "alerts_processed": 0,
470             "by_type": {},
471         }
472
473         all_alerts: list[NormalizedAlert] = []
474
475         for alert_type in alert_types:
476             type_count = 0
477
478             for state in states:
479                 try:

```



```

480         if alert_type == AlertType.CODE_SCANNING:
481             alerts = self.github.list_code_scanning_alerts(
org, repo, state)
482             for alert in alerts:
483                 normalized = self.normalizer.
normalize_code_scanning(org, repo, alert)
484                 all_alerts.append(normalized)
485                 type_count += 1
486
487             elif alert_type == AlertType.SECRET_SCANNING:
488                 alerts = self.github.
list_secret_scanning_alerts(org, repo, state)
489                 for alert in alerts:
490                     normalized = self.normalizer.
normalize_secret_scanning(org, repo, alert)
491                     all_alerts.append(normalized)
492                     type_count += 1
493
494             elif alert_type == AlertType.DEPENDABOT:
495                 alerts = self.github.list_dependabot_alerts(org
, repo, state)
496                 for alert in alerts:
497                     normalized = self.normalizer.
normalize_dependabot(org, repo, alert)
498                     all_alerts.append(normalized)
499                     type_count += 1
500
501             except requests.HTTPError as e:
502                 if e.response.status_code == 404:
503                     logger.info(f"{alert_type.value} not enabled
for {org}/{repo}")
504                 else:
505                     logger.error(f"Error fetching {alert_type.value
} for {org}/{repo}: {e}")
506
507             results["by_type"][alert_type.value] = type_count
508
509             # Batch upsert to ServiceNow
510             if all_alerts:
511                 sync_result = self.servicenow.upsert_alerts_batch(
all_alerts)
512                 results["sync_result"] = sync_result
513
514             results["alerts_processed"] = len(all_alerts)
515             return results
516
517     def reconcile_organization(
518         self,
519         org: str,
520         alert_types: Optional[list[AlertType]] = None,
521         max_repos: Optional[int] = None,
522     ) -> dict[str, Any]:
523         """Reconcile alerts for all repositories in an organization."""
524         results = {

```

```

525         "organization": org,
526         "timestamp": datetime.now(timezone.utc).isoformat(),
527         "repositories_processed": 0,
528         "total_alerts": 0,
529         "repositories": [],
530     }
531
532     repo_count = 0
533     for repo in self.github.list_org_repos(org):
534         if max_repos and repo_count >= max_repos:
535             break
536
537         repo_name = repo.get("name", "")
538         logger.info(f"Reconciling {org}/{repo_name}")
539
540         try:
541             repo_result = self.reconcile_repository(
542                 org, repo_name, alert_types
543             )
544             results["repositories"].append(repo_result)
545             results["total_alerts"] += repo_result["
alerts_processed"]
546         except Exception as e:
547             logger.error(f"Failed to reconcile {org}/{repo_name}: {
e}")
548
549             results["repositories"].append({
550                 "repository": f"{org}/{repo_name}",
551                 "error": str(e),
552             })
553
554             repo_count += 1
555
556         results["repositories_processed"] = repo_count
557     return results
558
559 def main():
560     """Main entry point."""
561     # Load configuration from environment
562     github_config = GitHubConfig(
563         token=os.environ["GITHUB_TOKEN"],
564     )
565
566     servicenow_config = ServiceNowConfig(
567         instance_url=os.environ["SERVICENOW_INSTANCE_URL"],
568         username=os.environ["SERVICENOW_USERNAME"],
569         password=os.environ["SERVICENOW_PASSWORD"],
570     )
571
572     # Initialize clients
573     github_client = GitHubClient(github_config)
574     servicenow_client = ServiceNowClient(servicenow_config)
575
576     # Create worker

```

```

577     worker = ReconciliationWorker(github_client, servicenow_client)
578
579     # Run reconciliation
580     org = os.environ.get("GITHUB_ORG", "")
581     repo = os.environ.get("GITHUB_REPO", "")
582
583     if repo:
584         # Single repository mode
585         result = worker.reconcile_repository(org, repo)
586     else:
587         # Organization mode
588         result = worker.reconcile_organization(org)
589
590     # Output results
591     print(json.dumps(result, indent=2, default=str))
592
593     # Exit with error if failures occurred
594     if result.get("sync_result", {}).get("failed", 0) > 0:
595         sys.exit(1)
596
597
598 if __name__ == "__main__":
599     main()

```

Listing 10: Python Reconciliation Worker with Full Feature Support

10 Implementation: GitHub Actions Integration

10.1 Security Results to ServiceNow DevOps

Leverage the official ServiceNow DevOps Security Results action:

```

1  name: Security Scanning with ServiceNow Integration
2
3  on:
4    push:
5      branches: [main, develop]
6    pull_request:
7      branches: [main]
8    schedule:
9      - cron: '0 6 * * 1' # Weekly Monday 6 AM UTC
10
11  permissions:
12    contents: read
13    security-events: write
14    actions: read
15
16  jobs:
17    # ===== CodeQL Analysis =====
18    codeql-analysis:
19      name: CodeQL SAST Scan
20      runs-on: ubuntu-latest
21

```

```

22     strategy:
23       fail-fast: false
24       matrix:
25         language: [javascript, python, go]
26
27     steps:
28       - name: Checkout repository
29         uses: actions/checkout@v4
30
31       - name: Initialize CodeQL
32         uses: github/codeql-action/init@v3
33         with:
34           languages: ${{ matrix.language }}
35           queries: +security-extended,security-and-quality
36
37       - name: Autobuild
38         uses: github/codeql-action/autobuild@v3
39
40       - name: Perform CodeQL Analysis
41         uses: github/codeql-action/analyze@v3
42         with:
43           category: "/language:${{ matrix.language }}"
44
45     # ===== Dependency Review =====
46     dependency-review:
47       name: Dependency Review
48       runs-on: ubuntu-latest
49       if: github.event_name == 'pull_request'
50
51     steps:
52       - name: Checkout repository
53         uses: actions/checkout@v4
54
55       - name: Dependency Review
56         uses: actions/dependency-review-action@v4
57         with:
58           fail-on-severity: high
59           deny-licenses: GPL-3.0, AGPL-3.0
60
61     # ===== Container Scanning =====
62     container-scan:
63       name: Container Security Scan
64       runs-on: ubuntu-latest
65       if: github.event_name == 'push'
66
67     steps:
68       - name: Checkout repository
69         uses: actions/checkout@v4
70
71       - name: Build container image
72         run: docker build -t app:${{ github.sha }} .
73
74       - name: Run Trivy vulnerability scanner
75         uses: aquasecurity/trivy-action@master

```

```

76         with:
77             image-ref: 'app:${{ github.sha }}'
78             format: 'sarif'
79             output: 'trivy-results.sarif'
80             severity: 'CRITICAL,HIGH,MEDIUM'
81
82     - name: Upload Trivy scan results to GitHub Security
83       uses: github/codeql-action/upload-sarif@v3
84       with:
85         sarif_file: 'trivy-results.sarif'
86         category: 'container-scanning'
87
88     # ===== ServiceNow Integration =====
89     servicenow-sync:
90       name: Sync to ServiceNow DevOps
91       runs-on: ubuntu-latest
92       needs: [codeql-analysis]
93       if: always() && github.event_name == 'push'
94
95       steps:
96         - name: Checkout repository
97           uses: actions/checkout@v4
98
99         - name: Register Security Results with ServiceNow
100          uses: ServiceNow/servicenow-devops-security-result@v3.1.0
101          with:
102            devops-integration-token: ${ secrets.
103            SN_DEVOPS_INTEGRATION_TOKEN }}
104            instance-url: ${ secrets.SN_INSTANCE_URL }}
105            tool-id: ${ secrets.SN_ORCHESTRATION_TOOL_ID }}
106            context-github: ${ toJSON(github) }}
107            job-name: 'ServiceNow Security Results'
108
109         - name: Notify on critical findings
110           if: failure()
111           uses: slackapi/slack-github-action@v1
112           with:
113             payload: |
114               {
115                 "text": ":rotating_light: Critical security findings
116                 detected in ${ github.repository }",
117                 "blocks": [
118                   {
119                     "type": "section",
120                     "text": {
121                       "type": "mrkdwn",
122                       "text": "*Security Alert*\nCritical findings
123                       detected in '${ github.repository }'\n<${ github.server_url }/${
124                       github.repository }/security|View Security Dashboard>"
125                     }
126                   }
127                 ]
128             }
129
130       env:

```

```

126         SLACK_WEBHOOK_URL: ${ secrets.SLACK_SECURITY_WEBHOOK }}
127
128 # ===== SBOM Generation =====
129 sbom-generation:
130   name: Generate and Upload SBOM
131   runs-on: ubuntu-latest
132   if: github.event_name == 'push' && github.ref == 'refs/heads/main'
133
134   steps:
135     - name: Checkout repository
136       uses: actions/checkout@v4
137
138     - name: Generate SBOM
139       uses: anchore/sbom-action@v0
140       with:
141         format: spdx-json
142         output-file: sbom.spdx.json
143
144     - name: Upload SBOM to ServiceNow
145       uses: ServiceNow/vulnerability-response@v2.0.1
146       with:
147         snSbomUser: ${ secrets.SN_SBOM_USERNAME }}
148         snSbomPassword: ${ secrets.SN_SBOM_PASSWORD }}
149         snInstanceUrl: ${ secrets.SN_INSTANCE_URL }}
150         ghToken: ${ secrets.GITHUB_TOKEN }}
151         ghAccountOwner: ${ github.repository_owner }}
152         repository: ${ github.event.repository.name }}
153         provider: 'repository'
154         path: 'sbom.spdx.json'

```

Listing 11: GitHub Actions: Complete Security Workflow with ServiceNow Integration

11 Automation Design: Playbooks and Workflows

11.1 Triage and Prioritization Matrix

Compute normalized priority using multiple signals:

Signal	Weighting and Logic
GitHub Severity	Direct mapping: critical=P1, high=P2, medium=P3, low=P4
CVSS Score	≥ 9.0 : P1, ≥ 7.0 : P2, ≥ 4.0 : P3, < 4.0 : P4
EPSS Score	Exploit Prediction Scoring: > 0.5 : escalate one level
Asset Criticality	Business-critical repo: escalate one level
Alert Type	Secret scanning: always P1 or P2 (credential exposure)
Push Protection Bypass	If push_protection_bypassed: true, immediate escalation
Known Exploited	If in CISA KEV catalog: immediate P1

Autofix Available Copilot Autofix available: may accelerate SLA

11.2 Assignment Resolution Strategy

Ownership resolution follows this precedence:

1. **CMDB/Application Registry:** Map repository to owning application/service via ServiceNow CMDB
2. **GitHub CODEOWNERS:** Parse `.github/CODEOWNERS` file for path-based ownership
3. **Repository Metadata:** Use GitHub repository topics or custom properties
4. **Recent Committers:** Assign to most active contributor if other methods fail
5. **Fallback Queue:** Route to AppSec triage queue for manual assignment

11.3 SLA Configuration Matrix

Priority	Secret	Code	Dependabot	Escalation
P1 / Critical	4 hours	24 hours	48 hours	Immediate page
P2 / High	24 hours	7 days	14 days	Daily summary
P3 / Medium	7 days	30 days	60 days	Weekly report
P4 / Low	30 days	90 days	180 days	Quarterly review

11.4 Exception Management Workflow

For dismissed alerts requiring risk acceptance:

1. **Documented Reason:** Mandatory dismissal reason (false positive, won't fix, used in tests, etc.)
2. **Risk Owner Approval:** Designated risk acceptance owner must approve
3. **Expiration Date:** All exceptions expire (default: 90 days for code, 30 days for secrets)
4. **Compensating Controls:** Document mitigations for accepted risk
5. **Review Cycle:** Automated reminder for review before expiration

12 Reliability Engineering

12.1 Idempotency Implementation

- **Delivery ID:** Store `X-GitHub-Delivery` UUID; reject duplicates
- **Canonical Key:** Use as record identity for upsert semantics
- **State Machine:** Only allow valid state transitions
- **Optimistic Locking:** Use `updated_at` timestamp for conflict detection

12.2 Retry and Dead-Letter Strategy

```
1  retry_policy:
2    initial_delay_ms: 1000
3    max_delay_ms: 300000 # 5 minutes
4    backoff_multiplier: 2.0
5    jitter_factor: 0.1
6    max_attempts: 5
7
8    # Retriable error codes
9    retrieable_status_codes:
10     - 429 # Rate limited
11     - 500 # Internal server error
12     - 502 # Bad gateway
13     - 503 # Service unavailable
14     - 504 # Gateway timeout
15
16  dlq_policy:
17    # Move to DLQ after max_attempts exhausted
18    max_receive_count: 5
19
20    # DLQ retention for investigation
21    retention_days: 14
22
23    # Alert thresholds
24    alert_on_dlq_depth: 10
25    alert_on_dlq_age_hours: 24
26
27    # Replay configuration
28    replay_batch_size: 100
29    replay_delay_ms: 500
30    require_manual_approval: true
```

Listing 12: Retry and DLQ Configuration

12.3 Observability Requirements

Key metrics to capture:

- **Ingest Latency:** GitHub webhook timestamp → ServiceNow record creation
- **Processing Success Rate:** By event type and action
- **Deduplication Hits:** Percentage of duplicate deliveries
- **Queue Depth:** Backlog size and age
- **API Rate Limit Consumption:** GitHub and ServiceNow
- **SLA Compliance:** By severity, alert type, and owning team
- **Mean Time to Remediation:** By severity and alert type
- **Autofix Acceptance Rate:** Copilot Autofix suggestions adopted

13 Security Hardening Checklist

1. Webhook Signature Verification

- Enforce `X-Hub-Signature-256` validation on every request
- Use constant-time comparison to prevent timing attacks
- Reject requests with missing or invalid signatures

2. Authentication and Authorization

- Use GitHub App tokens (not PATs) for API access
- Implement least-privilege permissions
- Rotate tokens regularly; automate rotation where possible
- Use OAuth 2.0 or mutual TLS for ServiceNow authentication

3. Secret Scanning Data Handling

- **CRITICAL:** Never log or store actual secret values
- Redact secrets in all stored payloads
- Retain only remediation-relevant metadata
- Implement data retention policies for alert data

4. Network Security

- Deploy webhook receivers behind WAF
- Implement rate limiting (per-IP and per-organization)
- Consider IP allow-listing using GitHub's Meta API
- Use TLS 1.3 for all connections

5. ServiceNow Security

- Restrict Scripted REST API with ACLs
- Use dedicated integration user with minimal permissions
- Enable audit logging for all security operations
- Implement input validation and sanitization

6. Change Management

- Version-control all mapping logic and workflow rules
- Use staged rollout with canary repositories
- Maintain rollback capability for configuration changes
- Document all integration touchpoints

14 Rollout Plan

14.1 Phase 0: Prerequisites (Week 1–2)

- Confirm GitHub Secret Protection and/or Code Security is enabled for target org/repos
- Install ServiceNow GitHub Application Vulnerability Integration
- Confirm SecOps licenses and roles are provisioned
- Define ownership mapping source of truth (CMDB vs CODEOWNERS)
- Establish SLA policies and escalation procedures
- Create GitHub App with required permissions

14.2 Phase 1: Baseline Import (Week 3–4)

- Enable official integration ingestion; verify records land correctly
- Validate severity mapping and deduplication logic
- Configure CWE/CVE enrichment
- Stand up initial dashboards and reports
- Establish baseline metrics for comparison

14.3 Phase 2: Webhook Event Path (Week 5–8)

- Deploy webhook receiver (external recommended)
- Implement signature verification and staging table
- Configure durable queue with DLQ
- Build Flow Designer workflows: assignment + SLA + notifications
- Test with canary repositories (non-critical)
- Monitor latency and error rates

14.4 Phase 3: Reconciliation and Advanced Features (Week 9–12)

- Deploy reconciliation poller with API rate limit awareness
- Implement drift detection: GitHub state vs ServiceNow state
- Add assignee synchronization (bidirectional if needed)
- Configure security campaign integration
- Add governance gates: false positive approvals, risk acceptance expiry
- Implement Copilot Autofix tracking and metrics

14.5 Phase 4: Production Hardening (Week 13–16)

- Performance testing under load (webhook storms)
- Security review and penetration testing
- Documentation and runbook completion
- Training for SecOps and development teams
- Full production rollout across organization
- Establish operational review cadence

15 Appendix A: Webhook Payload Examples

15.1 Code Scanning Alert Created (with Assignees)

```
1 {
2   "action": "created",
3   "alert": {
4     "number": 42,
5     "created_at": "2026-01-14T10:30:00Z",
6     "updated_at": "2026-01-14T10:30:00Z",
7     "url": "https://api.github.com/repos/acme/api/code-scanning/alerts/42",
8     "html_url": "https://github.com/acme/api/security/code-scanning/42"
9   },
10  "state": "open",
11  "fixed_at": null,
12  "dismissed_by": null,
13  "dismissed_at": null,
14  "dismissed_reason": null,
15  "dismissed_comment": null,
16  "assignees": [
17    {
18      "login": "jsmith",
19      "id": 12345,
20      "type": "User"
21    }
22  ],
23  "rule": {
24    "id": "js/sql-injection",
25    "severity": "error",
26    "security_severity_level": "critical",
27    "description": "SQL injection vulnerability",
28    "name": "SQL Injection",
29    "tags": ["security", "external/cwe/cwe-089"],
30    "help": {
31      "text": "Use parameterized queries..."
32    }
33  },
34  "tool": {
35    "name": "CodeQL",
```

```
35     "guid": null,
36     "version": "2.23.8"
37 },
38 "most_recent_instance": {
39     "ref": "refs/heads/main",
40     "state": "open",
41     "commit_sha": "abc123def456",
42     "location": {
43         "path": "src/db/queries.js",
44         "start_line": 42,
45         "end_line": 42,
46         "start_column": 10,
47         "end_column": 55
48     },
49     "message": {
50         "text": "User input flows to SQL query without sanitization"
51     }
52 }
53 },
54 "ref": "refs/heads/main",
55 "commit_oid": "abc123def456",
56 "repository": {
57     "id": 987654,
58     "name": "api",
59     "full_name": "acme/api",
60     "private": true,
61     "default_branch": "main"
62 },
63 "organization": {
64     "login": "acme",
65     "id": 111222
66 },
67 "sender": {
68     "login": "github-actions[bot]",
69     "type": "Bot"
70 }
71 }
```

Listing 13: Code Scanning Alert Webhook Payload

15.2 Secret Scanning Alert with Push Protection Bypass

```
1 {
2     "action": "created",
3     "alert": {
4         "number": 15,
5         "created_at": "2026-01-14T11:00:00Z",
6         "updated_at": "2026-01-14T11:00:00Z",
7         "url": "https://api.github.com/repos/acme/api/secret-scanning/
8         alerts/15",
9         "html_url": "https://github.com/acme/api/security/secret-scanning
10        /15",
```

```
9      "locations_url": "https://api.github.com/repos/acme/api/secret-scanning/alerts/15/locations",
10      "state": "open",
11      "secret_type": "aws_access_key_id",
12      "secret_type_display_name": "AWS Access Key ID",
13      "secret": "[REDACTED - never stored]",
14      "validity": "unknown",
15      "push_protection_bypassed": true,
16      "push_protection_bypassed_by": {
17        "login": "developer",
18        "id": 54321
19      },
20      "push_protection_bypassed_at": "2026-01-14T10:55:00Z",
21      "resolution": null,
22      "resolved_by": null,
23      "resolved_at": null,
24      "resolution_comment": null,
25      "assignees": [
26        {
27          "login": "security-team",
28          "id": 99999,
29          "type": "Team"
30        }
31      ]
32    },
33    "repository": {
34      "id": 987654,
35      "name": "api",
36      "full_name": "acme/api",
37      "private": true
38    },
39    "organization": {
40      "login": "acme",
41      "id": 111222
42    },
43    "sender": {
44      "login": "developer",
45      "type": "User"
46    }
47  }
```

Listing 14: Secret Scanning Alert Webhook Payload

16 Appendix B: Normalized Event Envelope

```
1  {
2    "delivery_id": "550e8400-e29b-41d4-a716-446655440000",
3    "event_type": "code_scanning_alert",
4    "action": "created",
5    "provider": "github",
6    "organization": "acme",
7    "repository": "acme/api",
```

```
8   "alert_type": "code_scanning",
9   "alert_number": 42,
10  "canonical_key": "github:acme/api:code_scanning:42",
11  "state": "open",
12  "severity": "critical",
13  "priority_computed": "P1",
14  "rule_id": "js/sql-injection",
15  "cwe_ids": ["CWE-089"],
16  "assignees": ["jsmith"],
17  "html_url": "https://github.com/acme/api/security/code-scanning/42",
18  "autofix_available": true,
19  "received_at": "2026-01-14T10:30:05Z",
20  "processing_state": "received",
21  "raw_payload_ref": "attachment_sys_id_or_hash"
22 }
```

Listing 15: Normalized Event Envelope for ServiceNow Staging

17 Appendix C: Primary References

17.1 GitHub Documentation

- Webhook events and payloads — <https://docs.github.com/en/webhooks/webhook-events-and-payloads>
- Validating webhook deliveries — <https://docs.github.com/en/webhooks/using-webhooks/validating-webhook-deliveries>
- Auditing security alerts — <https://docs.github.com/en/code-security/getting-started/auditing-security-alerts>
- REST API: Code scanning — <https://docs.github.com/en/rest/code-scanning/code-scanning>
- REST API: Secret scanning — <https://docs.github.com/en/rest/secret-scanning/secret-scanning>
- REST API: Dependabot alerts — <https://docs.github.com/en/rest/dependabot/alerts>
- GitHub Apps — <https://docs.github.com/en/apps>

17.2 GitHub Changelog (2025)

- GitHub Secret Protection and Code Security — <https://github.blog/changelog/2025-03-04-introducing-secret-protection-and-code-security/>
- Secret scanning alert assignees GA — <https://github.blog/changelog/2025-11-25-secret-scanning-alert-assignees-ga/>
- Code scanning alert assignees GA — <https://github.blog/changelog/2025-12-16-code-scanning-alert-assignees-ga/>

17.3 ServiceNow Documentation

- GitHub Application Vulnerability Integration — <https://store.servicenow.com/store/app/006dafa21b646a50a85b16db234bcba2>
- Vulnerability Response documentation — <https://www.servicenow.com/docs/bundle/security-management>

- ServiceNow DevOps Security Results action — <https://github.com/ServiceNow/servicenow-devops-security>
- ServiceNow Vulnerability Response action — <https://github.com/ServiceNow/vulnerability-response>

Document Version 2.0 — January 14, 2026
For questions or feedback, contact the AppSec Engineering team