Assignment 1.3: Bug Bounty Program Research & Scope Analysis

WAS600 – Web Security

Adeyemi Folarin

February 2026

Abstract

This document presents a comprehensive analysis of three bug bounty programs selected for hands-on security research: GitHub (HackerOne), Atlassian (Bugcrowd), and Yahoo (Intigriti). For each program, the analysis covers the platform context, program profile, bounty structure, historical background, target technology stack, and a detailed scope breakdown including in-scope assets, out-of-scope exclusions, allowed testing methods, and program-specific restrictions. A comparative analysis, scope compliance checklist, and tailored reconnaissance strategies per program are provided.

Contents

I Bug Bounty Program Research **4**

1 Introduction **4**

2 Platform Overview **4**
2.1 HackerOne . . . 4
2.2 Bugcrowd . . . 5
2.3 Intigriti . . . 6
2.4 Platform Comparison . . . 6

3 Program Selection Criteria **6**

II Program Profiles & Scope Analysis **7**

4 Program 1: GitHub (HackerOne) **7**
4.1 Program Overview . . . 7
4.2 Bounty Structure . . . 8
4.3 Historical Context . . . 8
4.4 Target Technology Profile . . . 9
4.5 In-Scope Assets . . . 9
4.6 Out-of-Scope Items . . . 10
4.7 Allowed Testing Methods and Restrictions . . . 10
4.8 How Scope Guides the Recon Approach . . . 11

5 Program 2: Atlassian (Bugcrowd) **11**
5.1 Program Overview . . . 11
5.2 Bounty Structure . . . 11
5.3 Historical Context . . . 12
5.4 Target Technology Profile . . . 12
5.5 In-Scope Assets . . . 13
5.6 Out-of-Scope Items . . . 13
5.7 Test Instance Setup Requirements . . . 14
5.8 Allowed Testing Methods and Restrictions . . . 14
5.9 How Scope Guides the Recon Approach . . . 15

6 Program 3: Yahoo (Intigriti) **15**
6.1 Program Overview . . . 15
6.2 Bounty Structure . . . 16
6.3 Historical Context . . . 16
6.4 Target Technology Profile . . . 16
6.5 In-Scope Assets . . . 17
6.6 Out-of-Scope Items . . . 17
6.7 Allowed Testing Methods and Restrictions . . . 18
6.8 How Scope Guides the Recon Approach . . . 19

III Comparative Analysis & Pre-Engagement Preparation 19

7 Program Comparison 19

8 Scope Compliance Checklist 20
8.1 Pre-Engagement Verification 20
8.2 During Recon 20
8.3 During Testing 21
8.4 Before Reporting 21

9 Recon Strategy per Program 21
9.1 GitHub: Depth-First Recon Strategy 22
9.2 Atlassian: Multi-Product Auth-Focused Strategy 22
9.3 Yahoo: Breadth-First Discovery Strategy 23

10 Conclusion 24

Part I
Bug Bounty Program Research

1 Introduction

Bug bounty programs are structured incentive programs operated by organizations to receive vulnerability reports from external security researchers. These programs serve a dual purpose: they provide organizations with continuous security testing beyond their internal teams, and they provide researchers with legal authorization and financial compensation for discovering and responsibly disclosing vulnerabilities.

The bug bounty ecosystem has matured significantly since the early programs launched by Mozilla (2004) and Google (2010). Today, thousands of organizations run bug bounty programs, and the market is facilitated by dedicated platforms that handle submission management, triage, communication, and payment processing.

This document analyzes three programs chosen to represent different platform ecosystems, scope characteristics, and target complexity levels:

1. **GitHub** on HackerOne – A large, well-funded program covering a complex developer tools platform with extensive API surface.

2. **Atlassian** on Bugcrowd – A medium-scope program covering enterprise collaboration tools with strict test-instance requirements.

3. **Yahoo** on Intigriti – A medium-large program with one of the broadest domain surfaces available, spanning legacy and modern web properties.

2 Platform Overview

The three major bug bounty platforms each have distinct characteristics that affect the researcher experience.

2.1 HackerOne

- **Founded:** 2012
- **Headquarters:** San Francisco, CA
- **Programs hosted:** 3,000+
- **Registered hackers:** 2,000,000+
- **Total bounties paid:** $300M+ (cumulative)

HackerOne is the largest bug bounty platform by program count and researcher base. Key platform features include:

- **Reputation system** – Researchers earn reputation points based on valid findings. Higher reputation unlocks private program invitations.

- **Signal metric** – Measures the ratio of valid to invalid reports, incentivizing quality submissions.

- **Mediation** – Platform-provided dispute resolution between researchers and programs.
- **Retesting** – Programs can request researchers to verify fixes, with additional compensation.
- **Pentest-as-a-Service** – Time-boxed engagements alongside the continuous bounty program.
- **Hacktivity feed** – Public disclosure of resolved reports (if both parties agree), providing learning opportunities.

Strengths: Largest researcher pool, most mature platform, extensive private program ecosystem, transparent reputation metrics.

Weaknesses: High competition on popular programs, some researchers report slow triage on certain programs.

2.2 Bugcrowd

- **Founded:** 2012
- **Headquarters:** San Francisco, CA
- **Programs hosted:** 1,000+
- **Registered hackers:** 500,000+
- **Total bounties paid:** $100M+ (cumulative)

Bugcrowd differentiates through its managed triage model:

- **Managed triage** – Bugcrowd's internal security team performs initial triage on all submissions before forwarding to the program. This reduces noise for programs but can add latency for researchers.
- **Vulnerability Rating Taxonomy (VRT)** – Standardized severity classification system that provides consistent expectations for bounty amounts.
- **Crowdsourced pen testing** – Time-boxed engagements with curated researcher groups.
- **Attack Surface Management** – Asset discovery services offered alongside bounty programs.
- **Kudos system** – Reputation and ranking mechanism.

Strengths: Managed triage provides faster initial response, VRT provides bounty predictability, strong enterprise program portfolio.

Weaknesses: Smaller researcher community than HackerOne, managed triage can occasionally misjudge severity.

2.3 Intigriti

- **Founded:** 2016
- **Headquarters:** Antwerp, Belgium
- **Programs hosted:** 300+
- **Registered hackers:** 100,000+
- **Total bounties paid:** $25M+ (cumulative)

Intigriti is the leading European bug bounty platform:

- **European focus** – Strong presence among European companies, with GDPR-aware program policies.
- **Managed triage** – Similar to Bugcrowd, Intigriti provides managed triage services.
- **Leaderboard system** – Quarterly and annual rankings with prizes.
- **Bonus promotions** – Programs frequently run time-limited bounty multipliers.
- **Researcher events** – Bug bounty live hacking events and conferences.
- **CVSS-based scoring** – Standardized severity assessment using CVSS v3.1.

Strengths: Less competition than HackerOne (smaller researcher pool), GDPR compliance built into platform, active community events.

Weaknesses: Fewer programs than HackerOne/Bugcrowd, newer platform with less historical data.

2.4 Platform Comparison

Feature	HackerOne	Bugcrowd	Intigriti
Founded	2012	2012	2016
Researcher base	2M+	500K+	100K+
Triage model	Program-managed (varies)	Managed by Bugcrowd	Managed by Intigriti
Reputation system	Reputation + Signal	Kudos + Ranking	Leaderboard
Payment methods	Bank, PayPal, Bitcoin	Bank, PayPal	Bank, PayPal
Public disclosure	Hacktivity (opt-in)	Limited	Limited
Regional focus	Global (US-heavy)	Global (US-heavy)	European focus

Table 1: Bug bounty platform comparison

3 Program Selection Criteria

The three programs were selected based on the following criteria:

1. **Platform diversity** – One program from each major platform (HackerOne, Bugcrowd, Intigriti) to experience different submission workflows, triage models, and platform features.

2. **Scope variety** – Programs with different scope characteristics:
   - GitHub: Deep, complex single-product scope with APIs and CI/CD
   - Atlassian: Multi-product scope with test-instance constraints
   - Yahoo: Broad domain surface with legacy and modern properties
3. **Technology stack diversity** – Each target runs different technologies:
   - GitHub: Ruby on Rails, Go, Node.js, React
   - Atlassian: Java, Kotlin, React, microservices architecture
   - Yahoo: PHP, Java, Node.js, diverse legacy stacks
4. **Bounty accessibility** – All three programs are public (no invitation required) and actively paying bounties, making them accessible for coursework and learning.
5. **Active response** – All three programs have demonstrated consistent response times and active engagement with researchers, reducing the risk of submissions going unreviewed.
6. **Learning value** – Each program presents distinct challenges that develop different security testing skills (API testing, multi-tenant isolation, legacy system analysis).

Part II
Program Profiles & Scope Analysis

Understanding and strictly adhering to a bug bounty program's scope is the single most important prerequisite before beginning any security testing. Scope defines the legal boundary between authorized security research and unauthorized access. Testing outside of scope can result in:

- **Legal consequences** – Unauthorized access to computer systems violates the Computer Fraud and Abuse Act (CFAA) in the United States and equivalent legislation in other jurisdictions.
- **Program disqualification** – Permanent ban from the platform and program.
- **Report rejection** – Findings on out-of-scope assets are not eligible for rewards.
- **Reputation damage** – Violations are tracked and may affect standing on other programs.

The scope information below was gathered directly from the official program policy pages. Scope details are subject to change; researchers must always verify current scope before testing.

4 Program 1: GitHub (HackerOne)

4.1 Program Overview

GitHub (owned by Microsoft since 2018) is the world's largest software development platform, hosting over 200 million repositories and serving over 100 million developers. The platform's bug bounty program is one of the most prestigious in the industry.

Attribute	Details
Program name	GitHub Security Bug Bounty
Platform	HackerOne
URL	https://hackerone.com/github
Launch year	2014
Program type	Public
Managed by	GitHub Security Team
Response SLA	First response within 5 business days
Resolution SLA	Varies by severity
Safe harbor	Yes

Table 2: GitHub program overview

4.2 Bounty Structure

GitHub uses a tiered bounty structure based on vulnerability severity:

Severity	Bounty Range	Example Vulnerabilities
Critical	$20,000 – $100,000+	Remote code execution, authentication bypass affecting all users, mass data exfiltration
High	$10,000 – $20,000	SQL injection, stored XSS on main application, SSRF with internal access
Medium	$4,000 – $10,000	IDOR with data exposure, reflected XSS, authorization bypass
Low	$617 – $4,000	Information disclosure, CSRF on sensitive actions, open redirect chains

Table 3: GitHub bounty tiers

GitHub's minimum bounty of $617 is among the highest in the industry, and the program has paid individual bounties exceeding $100,000 for critical findings.

4.3 Historical Context

Notable aspects of GitHub's security program history:

- **Total bounties paid:** Over $5 million since program launch.
- **Reports resolved:** 1,000+ valid vulnerabilities addressed.
- **Average time to first response:** Consistently within SLA (under 5 business days).
- **Notable past findings:** SAML authentication bypass (2020), GitHub Actions workflow injection patterns (2021–2022), GraphQL authorization issues (multiple).
- **GitHub Actions security:** The introduction of GitHub Actions in 2019 significantly expanded the attack surface and has been a source of numerous bounty-worthy findings, particularly around workflow injection (`pull_request_target` event), secret exfiltration, and runner escape.

- **Copilot expansion:** The addition of GitHub Copilot to the bounty scope in 2023 opened new AI-related vulnerability categories.

4.4 Target Technology Profile

- **Primary framework:** Ruby on Rails (main application)
- **Frontend:** React, TypeScript, Web Components
- **APIs:** REST API v3, GraphQL API v4
- **Infrastructure:** Azure (post-Microsoft acquisition), previously bare-metal
- **Git hosting:** Custom Git server implementation
- **CI/CD:** GitHub Actions (Node.js runners, containerized workflows)
- **CDN:** Fastly
- **Database:** MySQL (primary), Redis, Memcached
- **Search:** Elasticsearch
- **Authentication:** Session-based (web), OAuth (API), PAT (personal access tokens)

4.5 In-Scope Assets

GitHub's bug bounty program covers the following asset categories:

Asset Type	Assets
Web Application	`github.com` (primary), `*.github.com` subdomains
Web Application	`github.blog`, `githubstatus.com`
API	`api.github.com` (REST API v3)
API	GitHub GraphQL API
Desktop Application	GitHub Desktop (Windows, macOS)
Mobile Application	GitHub Mobile (iOS, Android)
Other	GitHub Actions (runner, workflow engine)
Other	GitHub Packages (npm, Docker, Maven registries)
Other	GitHub Pages (`*.github.io`)
Other	GitHub Codespaces
Other	GitHub Copilot
Source Code	Open-source GitHub repositories (`github.com/github/*`)

Table 4: GitHub in-scope assets

Key in-scope subdomains include:

- `github.com` – Main application
- `api.github.com` – REST and GraphQL API
- `gist.github.com` – Code snippet sharing
- `education.github.com` – Education program

- `docs.github.com` – Documentation
- `resources.github.com` – Marketing resources
- `*.githubusercontent.com` – User content CDN

4.6 Out-of-Scope Items

The following are explicitly excluded from the program:

Excluded domains and services:

- `enterprise.github.com` (GitHub Enterprise Cloud management)
- Third-party integrations and applications (GitHub Marketplace apps developed by third parties)
- Customer-hosted GitHub Enterprise Server instances
- `npm.community` and npm forums
- `electronjs.org` and Electron framework

Excluded vulnerability types:

- Social engineering or phishing attacks against GitHub employees
- Denial of service (DoS/DDoS) attacks
- Physical attacks against GitHub offices or data centers
- Spam or social engineering through GitHub features
- Vulnerabilities requiring physical access to a user's device
- Self-XSS (XSS that only affects the attacker's own session)
- Missing HTTP security headers without demonstrated impact
- Content injection without demonstrated security impact
- Rate limiting issues without security impact
- Clickjacking on pages with no sensitive actions
- Username/email enumeration via login or forgot password pages
- CSRF on logout or unauthenticated forms
- Vulnerabilities in outdated browsers or plugins
- Stack traces or path disclosures without demonstrated impact
- CSV injection
- Missing best practices in SSL/TLS configuration without working exploit
- Open redirects unless chained with another vulnerability

4.7 Allowed Testing Methods and Restrictions

- Automated scanning is permitted but must not degrade service availability.
- Rate limiting must be respected — excessive requests may trigger IP blocks.
- Researchers must use their own test accounts and repositories.
- Do not access, modify, or delete data belonging to other users.
- Do not test against production accounts of real users or organizations.
- Researchers should minimize data exposure and report immediately upon discovery.

Program rules:

1. **Safe harbor:** GitHub provides safe harbor for researchers acting in good faith within scope.

2. **Disclosure:** Do not disclose findings publicly until GitHub confirms the fix.
3. **No chaining for DoS:** Even if a vulnerability could be chained into a DoS, the DoS component must not be demonstrated.
4. **Data access:** If a vulnerability exposes user data, stop testing immediately and report. Do not download, store, or share exposed data beyond what is necessary to demonstrate the vulnerability.
5. **Accounts:** Only test with accounts you own. Do not attempt to access other users' accounts.

4.8 How Scope Guides the Recon Approach

GitHub's scope is broad, covering multiple web properties, APIs, desktop and mobile applications, and CI/CD infrastructure. This informs the recon strategy as follows:

1. **Subdomain enumeration** is critical given the `*.github.com` wildcard scope. Tools like `subfinder`, `amass`, and CT log scraping will yield a large number of in-scope subdomains.
2. **API testing** is a priority. Both the REST API (`api.github.com`) and GraphQL API are in scope, offering extensive attack surface for IDOR, authorization bypasses, and injection attacks.
3. **GitHub Actions** represents a complex attack surface involving workflow execution, secrets management, and runner security.
4. **Content at scale** — GitHub hosts billions of repositories, making content discovery and JavaScript analysis particularly relevant.
5. **Authentication flows** across web, API, desktop, and mobile clients provide multiple vectors for session management and authentication bypass testing.
6. Out-of-scope exclusions mean recon results must be filtered: any `enterprise.github.com` or third-party Marketplace findings should be tagged and excluded before active testing.

5 Program 2: Atlassian (Bugcrowd)

5.1 Program Overview

Atlassian is a global enterprise software company known for Jira (project management), Confluence (knowledge management), Bitbucket (source code hosting), Trello (kanban boards), and a suite of IT operations tools including Opsgenie and Statuspage. The company serves over 300,000 customers worldwide.

5.2 Bounty Structure

Atlassian's bounty structure follows Bugcrowd's Vulnerability Rating Taxonomy (VRT):

Attribute	Details
Program name	Atlassian Bug Bounty Program
Platform	Bugcrowd
URL	https://bugcrowd.com/atlassian
Launch year	2018
Program type	Public
Managed by	Atlassian Security Team + Bugcrowd Triage
Response SLA	First response within 3 business days (Bugcrowd triage)
Resolution SLA	Varies by product team
Safe harbor	Yes (via Bugcrowd terms)

Table 5: Atlassian program overview

Severity (VRT)	Bounty Range	Example Vulnerabilities
P1 (Critical)	$6,000 – $10,000+	RCE, authentication bypass, mass data access across tenants
P2 (High)	$3,000 – $6,000	Stored XSS, SQL injection, SSRF to internal services, privilege escalation
P3 (Medium)	$500 – $3,000	IDOR, reflected XSS, authorization issues between user roles
P4 (Low)	$100 – $500	Information disclosure, minor CSRF, low-impact misconfigurations

Table 6: Atlassian bounty tiers

5.3 Historical Context

- **Pre-bounty vulnerability history:** Atlassian products (particularly Jira and Confluence) have had significant CVEs over the years, including CVE-2022-26134 (Confluence OGNL injection, CVSS 9.8) and CVE-2023-22527 (Confluence template injection, CVSS 10.0). These historical vulnerabilities inform what types of issues to look for.
- **Cloud migration:** Atlassian's transition from Server/Data Center to Cloud has introduced new attack surface around multi-tenant isolation, API gateway security, and identity federation.
- **Atlassian Connect:** The app framework represents a significant attack surface for cross-tenant data access and permission escalation.
- **Acquisitions:** Atlassian's acquisitions (Trello in 2017, Opsgenie in 2018, Loom in 2023) each added new products to the bounty scope, sometimes with integration vulnerabilities between legacy and acquired codebases.

5.4 Target Technology Profile

- **Backend:** Java (Spring Boot), Kotlin, Python (some services)
- **Frontend:** React, TypeScript, Atlassian Design System

- **APIs:** REST APIs per product, GraphQL (newer services)
- **Infrastructure:** AWS (primary cloud provider)
- **Architecture:** Microservices, multi-tenant cloud
- **Authentication:** Atlassian Identity (`id.atlassian.com`), OAuth 2.0, API tokens
- **CDN:** CloudFront
- **Database:** PostgreSQL (primary), DynamoDB
- **App framework:** Atlassian Connect, Forge

5.5 In-Scope Assets

Asset Type	Assets
Web Application	Jira Cloud (`*.atlassian.net`)
Web Application	Confluence Cloud (`*.atlassian.net`)
Web Application	Bitbucket Cloud (`bitbucket.org`)
Web Application	Trello (`trello.com`)
Web Application	Statuspage (`*.statuspage.io`)
Web Application	Opsgenie (`*.opsgenie.com`)
Web Application	Atlassian Identity (`id.atlassian.com`)
API	Atlassian REST APIs for all cloud products
API	Atlassian Connect (app framework) APIs
Mobile	Jira Mobile, Confluence Mobile, Trello Mobile
Marketplace	Atlassian-developed Marketplace apps

Table 7: Atlassian in-scope assets

Important: Atlassian requires researchers to test against their own cloud instances. Free cloud instances can be created at `https://www.atlassian.com/try`.

5.6 Out-of-Scope Items

Excluded domains and services:

- `atlassian.com` corporate website (non-product pages)
- `community.atlassian.com` (community forums)
- `marketplace.atlassian.com` (third-party apps on Marketplace)
- Self-managed products (Jira Server, Confluence Server, Bitbucket Server)
- Data Center products
- `*.services.atlassian.com` (internal services)

Excluded vulnerability types:

- Denial of service attacks
- Social engineering or phishing
- Physical security issues
- Self-XSS without chaining
- Missing security headers without demonstrated exploit

- Clickjacking without sensitive action impact
- CSRF on logout
- Rate limiting or brute-force issues on non-authentication endpoints
- User enumeration via error messages
- Password complexity or account lockout policy issues
- Subdomain takeover without proof of exploit (must demonstrate control)
- Vulnerabilities requiring unlikely user interaction
- Issues in third-party Marketplace applications (report to app developer)
- Email SPF/DKIM/DMARC configuration issues

5.7 Test Instance Setup Requirements

Atlassian's program mandates testing on researcher-controlled instances:

1. **Create a free cloud site** at `https://www.atlassian.com/try` — this provides Jira, Confluence, and other products.
2. **Use your own Atlassian account** as the administrator.
3. **Create test users** within your instance for multi-user testing scenarios (e.g., IDOR testing between two users).
4. **Do not test on other customers' instances** — this is strictly prohibited and will result in program ban.
5. **Bitbucket testing** should use repositories within your own Bitbucket account.
6. **Trello testing** should use boards in your own workspace.

This requirement significantly shapes the recon approach: subdomain enumeration for `*.atlassian.net` will reveal thousands of customer instances, but only your own instance is in scope for active testing. Recon on the platform-level infrastructure (shared services, APIs, authentication) is permissible.

5.8 Allowed Testing Methods and Restrictions

1. **No production data access:** Do not attempt to access, modify, or exfiltrate data from production customer instances.
2. **Responsible disclosure:** 90-day disclosure window after fix confirmation.
3. **Duplicate policy:** First reporter receives the bounty. Bugcrowd's duplicate detection is based on root cause, not endpoint.
4. **Atlassian Connect apps:** Vulnerabilities in the Atlassian Connect framework are in scope; third-party apps built on Connect are not.
5. **Automated scanning:** Permitted on your own instances only. Do not run automated scanners against shared Atlassian infrastructure.
6. **Rate limits:** Respect API rate limits. Excessive requests will trigger blocks.

5.9 How Scope Guides the Recon Approach

1. **Instance isolation** is the defining constraint. Active testing (directory brute-forcing, parameter fuzzing, vulnerability scanning) must target your own `yoursite.atlassian.ne` instance.

2. **Passive recon on shared infrastructure** is permissible. CT log enumeration, DNS analysis, and technology fingerprinting of `*.atlassian.net` can reveal platform architecture.

3. **API-heavy testing** is prioritized. Atlassian products expose extensive REST APIs for every product feature, making API recon (endpoint discovery, parameter analysis, authentication testing) the primary attack vector.

4. **Multi-product interaction** — Jira, Confluence, Bitbucket, and Trello share authentication via `id.atlassian.com`. Cross-product authorization issues (e.g., accessing Confluence data via Jira API) are high-value targets.

5. **Marketplace app framework** testing examines how Atlassian Connect handles app permissions, data access, and cross-tenant isolation.

6. Subdomain enumeration results must be carefully filtered — customer instance subdomains (`customer.atlassian.net`) are out of scope for active testing.

6 Program 3: Yahoo (Intigriti)

6.1 Program Overview

Attribute	Details
Program name	Yahoo Bug Bounty Program
Platform	Intigriti
URL	https://app.intigriti.com/programs/yahoo
Launch year	Originally 2013 (migrated to Intigriti)
Program type	Public
Managed by	Yahoo Paranoids Security Team + Intigriti Triage
Response SLA	First response within 5 business days
Resolution SLA	Varies by product team
Safe harbor	Yes

Table 8: Yahoo program overview

Yahoo is one of the longest-standing internet companies, operating email, news, finance, sports, search, and advertising services. Now a subsidiary of Apollo Global Management (following the 2021 acquisition from Verizon), Yahoo continues to serve nearly a billion users globally. The “Paranoids” is Yahoo’s internal security team brand.

Severity	CVSS Range	Bounty Range	Examples
Critical	9.0–10.0	$10,000 – $15,000+	RCE, mass account takeover, authentication bypass
High	7.0–8.9	$4,000 – $10,000	SQL injection, stored XSS in Yahoo Mail, SSRF
Medium	4.0–6.9	$1,000 – $4,000	IDOR, reflected XSS, CSRF on sensitive actions
Low	0.1–3.9	$150 – $1,000	Information disclosure, minor misconfigurations

Table 9: Yahoo bounty tiers

6.2 Bounty Structure

Yahoo's bounty structure is CVSS-based, aligned with Intigriti's standard severity model:

Bonus promotions: Yahoo periodically runs bounty multiplier promotions. These have included 2x bounties for specific vulnerability types (e.g., authentication bypasses, RCE) or specific products (e.g., Yahoo Mail). Active promotions are listed on the program page and typically run for 1–4 weeks.

6.3 Historical Context

- **Major breaches:** Yahoo disclosed two massive data breaches — 500 million accounts (2014, disclosed 2016) and 3 billion accounts (2013, disclosed 2017). These breaches, among the largest in internet history, fundamentally shaped Yahoo's security posture and investment in bug bounty.
- **Program evolution:** Yahoo's bounty program started on HackerOne, moved to Bugcrowd, and is now on Intigriti. This migration history means older disclosed reports may be found across multiple platforms.
- **Verizon/Apollo ownership:** The ownership transitions (Yahoo → Verizon 2017 → Apollo 2021) have affected the program's scope and bounty levels.
- **Legacy infrastructure:** Yahoo's long history means significant legacy infrastructure — older PHP applications, legacy authentication systems, and deprecated but still accessible services.
- **Flickr and Tumblr:** These acquired properties have their own codebases and technology stacks, creating integration boundaries that are historically vulnerability-prone.

6.4 Target Technology Profile

- **Backend:** PHP (legacy), Java (core services), Node.js (newer services)
- **Frontend:** React (modern properties), legacy jQuery/vanilla JS
- **APIs:** REST APIs, some SOAP endpoints on legacy services

- **Infrastructure:** Hybrid (owned data centers + cloud)
- **Email:** Custom mail transfer infrastructure (Yahoo Mail)
- **Authentication:** Yahoo Account Key, OAuth 2.0, session cookies
- **CDN:** Akamai (primary), some Fastly usage
- **Advertising:** Custom ad-serving platform with JavaScript ad tags
- **Search:** Yahoo Search (powered by Bing partnership)
- **Flickr:** PHP/Node.js, separate authentication system
- **Tumblr:** PHP, separate authentication, third-party theme engine

6.5 In-Scope Assets

Asset Type	Assets
Web Application	`*.yahoo.com` (broad wildcard)
Web Application	`mail.yahoo.com` (Yahoo Mail)
Web Application	`finance.yahoo.com` (Yahoo Finance)
Web Application	`news.yahoo.com` (Yahoo News)
Web Application	`sports.yahoo.com` (Yahoo Sports)
Web Application	`search.yahoo.com` (Yahoo Search)
Web Application	`flickr.com` (photo sharing)
Web Application	`tumblr.com` (blogging platform)
API	Yahoo APIs across all products
Mobile	Yahoo Mail mobile apps (iOS, Android)
Mobile	Yahoo Finance mobile apps
Other	Yahoo advertising platform

Table 10: Yahoo in-scope assets

The `*.yahoo.com` wildcard scope makes this one of the broadest programs available, encompassing hundreds of subdomains across Yahoo's product portfolio.

6.6 Out-of-Scope Items

Excluded domains and services:

- `yahoo.co.jp` (Yahoo Japan – separate entity)
- Third-party integrations and plugins
- Yahoo-owned domains not listed in the in-scope section
- Content hosted by users (e.g., Tumblr blogs, Flickr photos)
- Yahoo advertising partner sites

Excluded vulnerability types:

- Denial of service (DoS/DDoS)
- Social engineering or phishing
- Physical attacks

- Self-XSS
- Vulnerabilities requiring MITM or physical access to device
- Missing HTTP headers without security impact
- Clickjacking without sensitive state-changing action
- Content spoofing or text injection without security impact
- CSRF on non-sensitive actions (e.g., logout, language change)
- Open redirects without chaining to another vulnerability
- Email-related issues (SPF, DKIM, DMARC misconfiguration)
- Descriptive error messages (stack traces, path disclosure)
- Software version disclosure
- Username enumeration
- Missing rate limits on non-critical endpoints
- Broken link hijacking
- Tabnabbing

6.7 Allowed Testing Methods and Restrictions

- Automated scanning is allowed but must be rate-limited to avoid service disruption.
- Researchers must use their own Yahoo accounts for testing.
- Do not access other users' data, emails, or accounts.
- Testing should focus on demonstrating impact without causing harm.
- Proof of concept should use minimal payloads (e.g., `alert(document.domain)` for XSS, not session hijacking).
- Researchers should not exfiltrate data beyond what is needed to prove the vulnerability.

Program rules:

1. **Coordinated disclosure:** Follow Intigriti's disclosure policy — no public disclosure until Yahoo confirms the fix or 90 days after reporting.
2. **Bonus promotions:** Yahoo occasionally runs time-limited bonus bounty promotions for specific vulnerability types. Check the program page for active promotions.
3. **Email access:** Yahoo Mail is in scope, but do not attempt to access other users' mailboxes. Test with your own accounts only.
4. **Financial data:** Yahoo Finance is in scope, but do not manipulate financial data or trading features in ways that could affect markets or other users.
5. **Duplicate handling:** Intigriti's duplicate policy awards the first valid report. Provide clear reproduction steps to speed triage.

6.8 How Scope Guides the Recon Approach

1. **Massive subdomain surface** — The `*.yahoo.com` wildcard means subdomain enumeration is the highest-priority recon phase. Yahoo has hundreds of active subdomains, many running different technology stacks and applications.

2. **Legacy systems** — Yahoo's long history means older subdomains may run legacy software with known vulnerabilities. Nuclei and Nikto scanning across all discovered subdomains is warranted.

3. **Multi-product diversity** — Mail, Finance, News, Sports, Search, Flickr, and Tumblr each represent distinct applications with different tech stacks, authentication models, and API surfaces.

4. **Flickr and Tumblr** as separate root domains (`flickr.com`, `tumblr.com`) require independent subdomain enumeration passes.

5. **Content-heavy applications** — News, Sports, and Finance rely on content management systems and real-time data feeds, which may expose injection vectors.

6. **Advertising platform** inclusion opens testing of ad-serving endpoints, tracking pixels, and JavaScript ad tags.

7. Due to the scale, prioritization is essential. Focus recon on high-value targets: Yahoo Mail (authentication, email parsing), Yahoo Finance (data integrity), and any subdomains running older or unusual software detected by technology fingerprinting.

Part III
Comparative Analysis & Pre-Engagement Preparation

7 Program Comparison

Key differences impacting recon strategy:

1. **GitHub** offers the highest bounties and broadest technology scope (web, API, CI/CD, desktop, mobile). Recon should be deep and thorough across all asset types.

2. **Atlassian** requires the most careful scope management due to the test-instance requirement. Passive recon on shared infrastructure is permissible, but active testing must be confined to researcher-owned instances.

3. **Yahoo** offers the widest domain surface area. Recon should prioritize breadth — discovering and fingerprinting as many subdomains as possible to find forgotten or under-maintained services.

Attribute	GitHub	Atlassian	Yahoo
Platform	HackerOne	Bugcrowd	Intigriti
Program age	2014 (12 yrs)	2018 (8 yrs)	2013 (13 yrs)
Min bounty	$617	$100	$150
Max bounty	$100,000+	$10,000+	$15,000+
Scope size	Large	Medium	Medium-Large
Wildcard domains	`*.github.com`	`*.atlassian.net` (own instances)	`*.yahoo.com`
Primary tech	Ruby on Rails	Java/Kotlin	PHP/Java
API surface	REST + GraphQL	REST (per product)	REST + legacy SOAP
Mobile in scope	Yes	Yes	Yes
Desktop in scope	Yes (GitHub Desktop)	No	No
Test account req.	Own GitHub account	Own cloud instance	Own Yahoo account
Automated scanning	Allowed (rate-limited)	Own instances only	Allowed (rate-limited)
DoS testing	Prohibited	Prohibited	Prohibited
Disclosure policy	Wait for fix	90-day window	90-day window
User base	100M+ developers	300K+ companies	1B+ users
Unique challenge	Actions/CI/CD security	Multi-tenant isolation	Legacy surface area
Competition level	Very high	High	Medium-high
Learning value	API & CI/CD testing	Multi-product auth	Legacy & breadth

Table 11: Side-by-side program comparison

8 Scope Compliance Checklist

Use this checklist before beginning any testing engagement.

8.1 Pre-Engagement Verification

- ☐ Read the complete program policy on the platform (HackerOne / Bugcrowd / Intigriti).
- ☐ Note the last-updated date of the scope — has it changed since your last review?
- ☐ Identify all in-scope domains and asset types.
- ☐ Identify all out-of-scope domains, subdomains, and asset types.
- ☐ Identify all excluded vulnerability types.
- ☐ Understand the program's disclosure policy.
- ☐ Create or verify test accounts on the target platform.
- ☐ For Atlassian: confirm your test cloud instance is active and accessible.

8.2 During Recon

- ☐ Cross-reference every discovered subdomain against the in-scope list before active testing.

- ☐ Tag and exclude out-of-scope subdomains from your target list.
- ☐ Verify that IP addresses resolved from subdomains belong to the target organization (not shared hosting with unrelated tenants).
- ☐ Set tool rate limits to avoid triggering WAF blocks or service degradation.
- ☐ Do not perform port scanning if the program prohibits it.
- ☐ Log all testing activity (timestamps, tools used, targets tested) for accountability.

8.3 During Testing

- ☐ Test only with accounts you own.
- ☐ Do not access, modify, or exfiltrate data belonging to other users.
- ☐ Use minimal-impact proof-of-concept payloads.
- ☐ Stop testing and report immediately upon discovering exposed user data.
- ☐ Do not chain vulnerabilities into denial-of-service conditions.
- ☐ Do not publicly disclose findings before program authorization.

8.4 Before Reporting

- ☐ Confirm the affected asset is in scope.
- ☐ Confirm the vulnerability type is not excluded.
- ☐ Verify reproduction steps work consistently.
- ☐ Check the program's disclosed issues for potential duplicates.
- ☐ Prepare clear documentation with steps, screenshots, and impact assessment.
- ☐ Remove any sensitive data (tokens, PII) from your report that is not essential to the proof of concept.

9 Recon Strategy per Program

The reconnaissance methodology from Assignment 1.2 must be adapted to each program's unique characteristics. This section outlines how the general methodology maps to each target.

9.1 GitHub: Depth-First Recon Strategy

GitHub's scope rewards deep analysis of a complex, well-maintained application rather than breadth-based asset discovery.

Phase priorities:

1. **Passive recon (High priority):**
   - GitHub's own public repositories are a goldmine. Search `github/github` (if accessible), configuration repositories, and engineering blog posts.
   - CT log enumeration to discover internal/staging subdomains.
   - Google dorking focused on `site:github.com` with API and configuration paths.
2. **Subdomain discovery (Medium priority):**
   - Enumerate `*.github.com` and `*.githubusercontent.com`.
   - Focus on validating which subdomains run distinct applications vs. redirects.
3. **API recon (Highest priority):**
   - Map the REST API v3 systematically (GitHub publishes API docs at `docs.github.com/en/rest`).
   - Enumerate GraphQL schema using introspection queries.
   - Test authentication boundaries: PAT vs. OAuth vs. GitHub App permissions.
   - Focus on IDOR patterns across organization, repository, and user contexts.
4. **GitHub Actions recon (High priority):**
   - Study workflow injection patterns (`pull_request_target`).
   - Analyze how secrets are exposed to workflow steps.
   - Test runner isolation and container escape scenarios.
5. **Vulnerability scanning (Medium priority):**
   - Nuclei with GitHub-specific templates.
   - Focus on information disclosure and misconfiguration rather than injection (GitHub's main app is well-hardened).

Expected high-value targets: GraphQL API authorization issues, GitHub Actions workflow injection, OAuth scope escalation, cross-repository data access via APIs, GitHub Copilot prompt injection.

9.2 Atlassian: Multi-Product Auth-Focused Strategy

Atlassian's scope centers on cloud product instances that the researcher controls, making authorization and authentication testing the primary focus.

Phase priorities:

1. **Instance setup (Prerequisites):**
   - Create a free Atlassian Cloud instance with Jira, Confluence, and Bitbucket.
   - Create at least 3 test user accounts with different permission levels (admin, standard user, guest).

- Install several Atlassian Connect apps from the Marketplace.

2. **Passive recon on shared infrastructure (Medium priority):**
   - CT log enumeration of `*.atlassian.net` and `*.atlassian.com`.
   - Technology fingerprinting of shared API gateways and authentication endpoints.
   - Review Atlassian's public developer documentation for API patterns.
3. **API recon (Highest priority):**
   - Map Jira REST API endpoints (issue CRUD, permissions, workflows).
   - Map Confluence REST API (pages, spaces, permissions).
   - Test cross-product API access (can a Jira API token access Confluence data?).
   - Test Atlassian Connect app permission boundaries.
4. **Authorization testing (Highest priority):**
   - IDOR testing between user accounts with different privilege levels.
   - Test tenant isolation — can API calls from one instance access another?
   - Test Atlassian Identity SSO flows for authorization bypass.
   - Test role-based access control across Jira projects and Confluence spaces.
5. **Content discovery (Low priority):**
   - Directory brute-forcing on your own instance only.
   - Focus on admin paths, debug endpoints, and API documentation.

Expected high-value targets: Cross-tenant data access, privilege escalation between user roles, Atlassian Connect app permission bypass, IDOR in Jira issue and Confluence page access, identity federation bypass between products.

9.3 Yahoo: Breadth-First Discovery Strategy

Yahoo's vast domain surface and legacy infrastructure reward a breadth-first recon approach focused on discovering forgotten or under-maintained assets.

Phase priorities:

1. **Subdomain discovery (Highest priority):**
   - Aggressive enumeration of `*.yahoo.com`, `*.flickr.com`, and `*.tumblr.com`.
   - Use all available tools: subfinder, amass, assetfinder, CT logs, waybackurls.
   - Expect hundreds of unique subdomains.
   - Prioritize subdomains running non-standard technologies or older software versions.
2. **Technology fingerprinting (High priority):**
   - Run httpx with `-tech-detect` across all discovered subdomains.
   - Identify legacy PHP applications, outdated frameworks, and non-standard services.
   - Flag subdomains with interesting technology combinations (e.g., PHP + old jQuery).
3. **Vulnerability scanning (High priority):**

- Run Nuclei across all live hosts with broad template sets.
- Focus on known CVE templates for detected technologies.
- Nikto scanning on hosts running older web servers.

4. **Wayback Machine recon (High priority):**
   - Yahoo's long history means extensive Wayback Machine coverage.
   - Search for removed APIs, old admin interfaces, and deprecated endpoints.
   - Identify parameters and paths no longer linked from current navigation.

5. **Yahoo Mail specific (High priority):**
   - Email parsing vulnerabilities (HTML injection in email rendering).
   - Attachment handling issues.
   - Mail forwarding and filter bypass.
   - Authentication and session management.

Expected high-value targets: Legacy subdomain vulnerabilities, Yahoo Mail rendering bugs (stored XSS, content injection), forgotten admin panels and debug endpoints, SSRF via advertising or content-fetching features, authentication issues on older properties.

10 Conclusion

The three selected programs provide a comprehensive bug bounty research experience:

1. **GitHub** develops skills in API security testing, CI/CD attack surface analysis, and deep application logic testing. The high bounties and active community make it rewarding despite fierce competition.

2. **Atlassian** develops skills in multi-tenant security, authorization testing, and enterprise SaaS attack patterns. The test-instance requirement provides a safe sandbox for aggressive testing techniques.

3. **Yahoo** develops skills in large-scale reconnaissance, legacy system analysis, and email security. The broad scope offers the highest probability of discovering overlooked assets.

Together, these programs cover the full spectrum of modern bug bounty hunting: deep analysis of complex applications (GitHub), controlled multi-product testing (Atlassian), and wide-scope asset discovery (Yahoo). The reconnaissance methodology from Assignment 1.2 adapts to each target, with phase priorities shifting based on the program's unique characteristics.

The scope compliance checklist and per-program recon strategies in this document serve as the operational framework for Assignment 1.4, where the methodology will be executed against each target and findings documented.