COALFIRE CONTROLS

Report on GitHub, Inc.’s Description of Its Information Technology General Control System for the GitHub Enterprise Cloud service and on the Suitability of the Design and Operating Effectiveness of Its Controls Throughout the Period October 1, 2024 to March 31, 2025

SOC 1® - SOC for Service Organizations: ICFR - Integrated Type 2 Report Prepared in Accordance with the AICPA SSAE No. 18 and IAASB ISAE 3402 Standards

Table of Contents

Section 1

Independent Service Auditor’s Report

Independent Service Auditor's Report

To: GitHub, Inc. ("GitHub")

Scope

We have examined GitHub's description of its information technology general control system for the GitHub Enterprise Cloud service throughout the period October 1, 2024 to March 31, 2025 (description), and the suitability of the design and operating effectiveness of the controls included in the description to achieve the related control objectives stated in the description, based on the criteria identified in management of GitHub's assertion. The controls and control objectives included in the description are those that management of GitHub believes are likely to be relevant to user entities' internal control over financial reporting, and the description does not include those aspects of the information technology general control system for the GitHub Enterprise Cloud service that are not likely to be relevant to user entities' internal control over financial reporting.

The description indicates that certain control objectives specified in the description can be achieved only if complementary user entity controls assumed in the design of GitHub's controls are suitably designed and operating effectively, along with related controls at the service organization. Our examination did not extend to such complementary user entity controls, and we have not evaluated the suitability of the design or operating effectiveness of such complementary user entity controls.

GitHub uses subservice organizations for data center colocation and infrastructure hosting services. The description includes only the control objectives and related controls of GitHub and excludes the control objectives and related controls of the subservice organizations. The description also indicates that certain control objectives specified by GitHub can be achieved only if complementary subservice organization controls assumed in the design of GitHub's controls are suitably designed and operating effectively, along with the related controls at GitHub. Our examination did not extend to controls of the subservice organizations, and we have not evaluated the suitability of the design or operating effectiveness of such complementary subservice organization controls.

The information included in Section 5, "Other Information Provided by GitHub, Inc.", is presented by management of GitHub to provide additional information and is not a part of GitHub's description of its information technology general control system for the GitHub Enterprise Cloud service made available to user entities during the period October 1, 2024 to March 31, 2025. Information included in management's response to identified exceptions has not been subjected to the procedures applied in the examination of the description of the information technology general control system for the GitHub Enterprise Cloud service and of the suitability of the design and operating effectiveness of controls to achieve the related control objectives stated in the description of the information technology general control system for the GitHub Enterprise Cloud service and, accordingly, we express no opinion on it.

Service Organization's Responsibilities

In Section 2 of this report, GitHub has provided an assertion about the fairness of the presentation of the description and suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description. Management of GitHub is responsible for preparing the description and assertion, including the completeness, accuracy, and method of presentation of the description and assertion, providing the services covered by the description, specifying the control objectives and stating them in the description, identifying the risks that threaten the achievement of the

control objectives, selecting the criteria stated in the assertion, and designing, implementing, and documenting controls that are suitably designed and operating effectively to achieve the related control objectives stated in the description.

Service Auditor's Responsibilities

Our responsibility is to express an opinion on the fairness of the presentation of the description and on the suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description, based on our examination.

Our examination was conducted in accordance with attestation standards established by the American Institute of Certified Public Accountants and International Standard on Assurance Engagements (ISAE) 3402, *Assurance Reports on Controls at a Service Organization*, issued by the International Auditing and Assurance Standards Board (IAASB). Those standards require that we plan and perform the examination to obtain reasonable assurance about whether, in all material respects, based on the criteria in management's assertion, the description is fairly presented and the controls were suitably designed and operating effectively to achieve the related control objectives stated in the description throughout the period October 1, 2024 to March 31, 2025. We believe that the evidence we obtained is sufficient and appropriate to provide a reasonable basis for our opinion.

An examination of a description of a service organization's system and the suitability of the design and operating effectiveness of controls involves –

- Performing procedures to obtain evidence about the fairness of the presentation of the description and the suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description, based on the criteria in management's assertion.
- Assessing the risks that the description is not fairly presented and that the controls were not suitably designed or operating effectively to achieve the related control objectives stated in the description.
- Testing the operating effectiveness of those controls that management considers necessary to provide reasonable assurance that the related control objectives stated in the description were achieved.
- Evaluating the overall presentation of the description, suitability of the control objectives stated in the description, and suitability of the criteria specified by the service organization in its assertion.

Service Auditor's Independence and Quality Control

We are required to be independent and to meet our other ethical responsibilities in accordance with relevant ethical requirements in the United States of America relating to the examination engagement. We have complied with those requirements.

We have also applied the Statements on Quality Control Standards established by the AICPA and the International Standards on Quality Management issued by the IAASB and, accordingly, maintain a comprehensive system of quality control.

Inherent Limitations

The description is prepared to meet the common needs of a broad range of user entities and their auditors who audit and report on user entities' financial statements and may not, therefore, include every aspect of the system that each individual user entity may consider important in its own particular environment. Because of their nature, controls at a service organization may not prevent, or detect and correct, all

misstatements in its information technology general control system. Also, the projection to the future of any evaluation of the fairness of the presentation of the description, or conclusions about the suitability of the design or operating effectiveness of the controls to achieve the related control objectives, is subject to the risk that controls at a service organization may become ineffective.

Description of Tests of Controls

The specific controls tested and the nature, timing, and results of those tests are listed in Section 4 of this report.

Opinion

In our opinion, in all material respects, based on the criteria described in management of GitHub's assertion–

a. The description fairly presents the information technology general control system for the GitHub Enterprise Cloud service that was designed and implemented throughout the period October 1, 2024 to March 31, 2025.

b. The controls related to the control objectives stated in the description were suitably designed to provide reasonable assurance that the control objectives would be achieved if the controls operated effectively throughout the period October 1, 2024 to March 31, 2025, and subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout the period October 1, 2024 to March 31, 2025.

c. The controls operated effectively to provide reasonable assurance that the control objectives stated in the description were achieved throughout the period October 1, 2024 to March 31, 2025 if complementary subservice organization and user entity controls assumed in the design of GitHub's controls operated effectively throughout the period October 1, 2024 to March 31, 2025.

Restricted Use

This report, including the description of tests of controls and results thereof in Section 4, is intended solely for the information and use of GitHub, user entities of GitHub's information technology general control system for the GitHub Enterprise Cloud service during some or all of the period October 1, 2024 to March 31, 2025, and their auditors who audit and report on such user entities' financial statements or internal control over financial reporting and have a sufficient understanding to consider it, along with other information, including information about controls implemented by user entities themselves, when assessing the risks of material misstatement of user entities' financial statements. This report is not intended to be, and should not be, used by anyone other than these specified parties.

If a report recipient is not a specified party as defined above and has obtained this report, or has access to it, use of this report is the non-specified user's sole responsibility and at the non-specified user's sole and exclusive risk. Non-specified users may not rely on this report and do not acquire any rights against Coalfire Controls, LLC as a result of such access. Further, Coalfire Controls, LLC does not assume any duties or obligations to any non-specified user who obtains this report and/or has access to it.

Coalfire Controls LLC

Greenwood Village, Colorado
June 2, 2025

Section 2

Assertion of the Management of GitHub, Inc.

GitHub

88 Colin P Kelly Jr Street,
San Francisco, CA 94107
Tel: 415-448-6673 (main)

Assertion of the Management of GitHub, Inc. (“GitHub”)

We have prepared the description of GitHub’s information technology general control system for the GitHub Enterprise Cloud service throughout the period October 1, 2024 to March 31, 2025 (description) for user entities of the system during some or all of the period October 1, 2024 to March 31, 2025, and their auditors who audit and report on such user entities’ financial statements or internal control over financial reporting and have a sufficient understanding to consider it, along with other information, including information about controls implemented by subservice organizations and user entities of the system themselves when assessing the risks of material misstatement of user entities’ financial statements.

GitHub uses subservice organizations for data center colocation and infrastructure hosting services. The description includes only the control objectives and related controls of GitHub and excludes the control objectives and related controls of the subservice organizations. The description also indicates that certain control objectives specified in the description can be achieved only if complementary subservice organization controls assumed in the design of our controls are suitably designed and operating effectively, along with the related controls. The description does not extend to controls of the subservice organizations.

The description indicates that certain control objectives specified in the description can be achieved only if complementary user entity controls assumed in the design of GitHub’s controls are suitably designed and operating effectively, along with related controls at the service organization. The description does not extend to controls of the user entities.

We confirm, to the best of our knowledge and belief, that:

1) The description fairly presents GitHub’s information technology general control system made available to user entities of the system during some or all of the period October 1, 2024 to March 31, 2025 as it relates to controls that are likely to be relevant to user entities’ internal control over financial reporting. The criteria we used in making this assertion were that the description:
    a) Presents how the system made available to user entities of the system was designed and implemented including, if applicable:
        i) The types of services provided.
        ii) The procedures, within both automated and manual systems, by which requests for those services are provided, including, as appropriate, procedures by which services are initiated, authorized, recorded, processed, corrected as necessary, and transferred to the reports and other information prepared for user entities of the system.
        iii) How the system captures and addresses significant events and conditions.

iv) The process used to prepare reports and other information for user entities.

v) The services performed by subservice organizations, and that the carve-out method has been used in relation to them.

vi) The specified control objectives and controls designed to achieve those objectives including, as applicable, complementary user entity controls and complementary subservice organization controls assumed in the design of the controls.

vii) Other aspects of our control environment, risk assessment process, information and communications, control activities, and monitoring activities that are relevant to the services provided.

b) Includes relevant details of changes to GitHub's information technology general control system during the period covered by the description.

c) Does not omit or distort information relevant to the system, while acknowledging that the description is prepared to meet the common needs of a broad range of user entities and their user auditors and may not, therefore, include every aspect of GitHub's information technology general control system that each individual user entity of the system and its auditor may consider important in its own particular environment.

2) The controls related to the control objectives stated in the description were suitably designed and operating effectively throughout the period October 1, 2024 to March 31, 2025 to achieve those control objectives if subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout the period October 1, 2024 to March 31, 2025. The criteria we used in making this assertion were that:

a) The risks that threaten the achievement of the control objectives stated in the description have been identified by management of the service organization.

b) The controls identified in the description would, if operating effectively, provide reasonable assurance that those risks would not prevent the control objectives stated in the description from being achieved.

c) The controls were consistently applied as designed, including whether manual controls were applied by individuals who have the appropriate competence and authority.

GitHub, Inc.

Section 3

Description of GitHub, Inc.’s Information Technology General Control System for the GitHub Enterprise Cloud service Throughout the Period October 1, 2024 to March 31, 2025

Type of Services Provided

GitHub (or “the Company”) is an independently operated subsidiary of Microsoft and generated its first commit in 2007. It is headquartered in San Francisco, CA, with additional offices in Bellevue, WA, and Oxford, United Kingdom. GitHub currently employs approximately 3,000 employees, with approximately 95 percent of the workforce remote.

GitHub is a web-based software development platform built on the Git version control software. Primarily used for software code, GitHub offers the distributed version control and source code management functionality of Git with additional features and enhancements. Specifically, it provides access control and several collaboration features, including bug tracking, feature requests, task management, GitHub Teams, pull requests, discussions, issues, pages, projects, docs, and wikis. GitHub Enterprise Cloud service with Data Residency is an optional version of GitHub Enterprise Cloud service that offers features that provide customers control over the location where their data is stored. Customers can optionally add Copilot Business, Copilot Enterprise, Actions, and GitHub Advanced Security. In this report, unless specified otherwise, statements about GitHub Enterprise Cloud service apply to all in-scope products represented above.

The following are descriptions of GitHub Enterprise Cloud service features.

Organizations

An organization is a collection of user accounts that owns repositories. Organizations have one or more owners, who have administrative privileges for the organization. When a user creates an organization, it does not have any repositories associated with it. At any time, members of the organization with the owner role can add new repositories or transfer existing repositories.

Code Hosting

GitHub is one of the largest code hosts in the world, with millions of projects. Private, public, or open-source repositories are equipped with tools to host, version, and release code. Unlimited private repositories allow keeping the code in one place, even when using Subversion or working with large files using Git Large File Storage.

Changes can be made to code in precise commits, allowing for quick searches on commit messages in the revision history to find a change. In addition, blame view enables users to trace changes and discover how the file, and code base have evolved.

With sharing, changes can be packaged from a recently closed milestone or finished project into a new release. Users can draft and publish release notes, publish pre-release versions, attach files, and link directly to the latest download.

Code Management

Code review is a critical path to better code and is fundamental to how GitHub works. Built-in review tools make code review an essential part of team development workflows.

A pull request is a living conversation where ideas can be shared, tasks assigned, details discussed, and reviews conducted. Reviews happen faster when GitHub shows a user exactly what has changed. Diffs compare versions of source code side by side, highlighting the parts that are new or have been edited or deleted.

Pull requests also enable clear feedback, review requests, and comments in context with comment threads within the code. Comments may be bundled into one review or in reply to comments inline as a conversation.

GitHub allows customers to protect important branches by setting branch protection rules, which define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history. Protected branches allow for better quality code management. Repositories can be configured to require status checks, such as continuous integration (CI) tests, reducing both human error and administrative overhead.

Project Management

Project boards allow users to reference every issue and pull request in a card, providing a drag-and-droppable snapshot of the work that teams do in a repository. This feature can also function as an agile idea board to capture early ideas that come up as part of a standup or team sync, without polluting the issues.

Issues enable team task tracking, with resources identified and tasks assigned within a team. Issues may be used to track a bug, discuss an idea with an @ mention, or start distributing work. Issue and pull request assignments to one or more teammates make it clear who is doing what work and what feedback and approvals have been requested.

Milestones can be added to issues or pull requests to organize and track progress on groups of issues or pull requests in a repository.

Teams (User Management)

NOTE: This section refers to teams, not the plan called GitHub Team.

Building software is as much about managing teams and communities as it is about code. Users set roles and expectations without starting from scratch. Customized common codes of conduct can be created for any project, with pre-written licenses available right from the repository.

Teams enable the ability to organize people, provide level-up access with administrative roles, and tune permissions for nested teams. Discussion threads keep conversations on topic using moderation tools, such as issue and pull-request locking, to help teams stay focused on code. For maintaining open-source projects, user blocking reduces noise and keeps conversations productive.

Documentation

GitHub allows documentation to be created and maintained in any repository, and wikis are available to create documentation with version control. Each wiki is its own repository, so every change is versioned and comparable. With a text editor, users can add documents in the text formatting language of choice, such as Textile or GitHub Flavored Markdown.

GitHub Enterprise Cloud service with Data Residency

GitHub Enterprise Cloud service with Data Residency is GitHub Enterprise Cloud service's platform with robust data residency features for enterprises. With improved enterprise-grade features and more control over where code is stored, GitHub Enterprise Cloud service with Data Residency will help more enterprise customers meet their security and compliance needs and addresses:

- The ability to store code and repository data in a preferred region

- Enhanced user control, allowing organizations to manage and control user accounts
- Unique namespaces specific to a company on ghe.com isolated from the open-source community
- Enhanced availability and support for zone-based business continuity and disaster recovery

GitHub Copilot

GitHub Copilot is an artificial intelligence (AI)-powered coding assistant that helps developers write code faster. GitHub Copilot (including the Copilot for Non– GitHub Enterprise Cloud service customers product offering) is available through:

- GitHub Copilot Business for GitHub organization or enterprise accounts, which gives control over Copilot policies, including which members can use Copilot
- GitHub Copilot Enterprise for enterprise accounts on GitHub Enterprise Cloud service, which includes all Copilot Business features along with additional AI features on GitHub. With this subscription plan users can choose to assign either Copilot Enterprise or Copilot Business to each individual organization in the enterprise

GitHub Copilot features include, but are not limited to, the following:

Copilot Feature	Short Description
Code Completion	Autocomplete-style suggestions from Copilot in supported Integrated Development Environments (IDEs) (Visual Studio Code, Visual Studio, JetBrains IDEs, Azure Data Studio, and Vim/Neovim).
Copilot Chat	A chat interface that lets users ask coding-related questions. GitHub Copilot Chat is available on the GitHub website, in GitHub Mobile, and in supported IDEs (Visual Studio Code, Visual Studio, and JetBrains IDEs). Users can also use skills with Copilot Chat.
Copilot in the command line interface (CLI)	A chat-like interface in the terminal, where users can ask questions about the command line. Users can ask Copilot to provide command suggestions or explanations of commands.
Copilot pull request summaries	AI-generated summaries of the changes that were made in a pull request, which files they impact, and what a reviewer should focus on when they conduct their review.
Copilot Extensions	Enables developers to build and deploy to the cloud with their preferred tools and services from the GitHub Marketplace or through private tooling created by organizations. Extensions are supported in GitHub Copilot Chat on GitHub.com, in Visual Studio, and in Visual Studio Code.
Code referencing	Grants the option to notify developers in situations where Copilot produces suggestions of 150 characters or more that match public code, including details about the matches found, the repositories the code was found in, and potential licenses detected. Additionally, GitHub's indemnity extends to dev teams use of code referencing where GitHub Copilot Business or GitHub Copilot Enterprise customers comply with cited licenses.
Code reviews	Allows developers to configure Copilot as a reviewer on a repository to automatically request a code review and feedback from Copilot for all new pull requests.

Copilot Feature	Short Description
Metrics application programming interface (API)	Supplies information about Copilot's usage within GitHub enterprise, organizations, and teams, offering visibility into utilization of individual Copilot features and the volume of daily active users.
Copilot text completion (beta) (Copilot Enterprise only)	AI-generated text completion to help you write pull request descriptions quickly and accurately.
Fine-tuned models (beta)(Copilot Enterprise Only)	Enables enterprise users the ability to customize Copilot with their proprietary codebases and coding practices, allowing the delivery of code suggestions that are not just syntactically correct, but more deeply aligned with the team's coding style and standards.
Copilot knowledge bases (Copilot Enterprise only)	Create and manage collections of documentation to use as context for chatting with Copilot. When users ask a question in Copilot Chat on GitHub.com or in Visual Studio Code, users can specify a knowledge base as the context for their question.

GitHub Copilot Administrator features include, but are not limited to, the following.

Copilot Feature	Short Description
Policy management	Manage policies for Copilot in the user's organization or enterprise.
Access management	Enterprise owners can specify which organizations in the enterprise can use Copilot, and organization owners can specify which organization members can use Copilot.
Usage data	Review Copilot usage data within an organization or enterprise to inform how to manage access and drive adoption of Copilot.
Audit logs	Review audit logs for Copilot in an organization to understand what actions have been taken and by which users.
Exclude files and content	Configure Copilot to ignore certain files and content. This can be useful if users have data that they do not want to be available to Copilot.

GitHub Actions

GitHub Actions automates CI/continuous delivery (CD) software workflows by enabling the build, test, and deployment of code directly from GitHub, with code reviews, branch management, and issue triaging customized to work the way developers need.

GitHub Actions initiates workflows for events such as push, issue creation, or a new release, and actions can be combined and configured for the services used, built, and maintained by the community.

GitHub Actions supports additional options to build containers, deploy web services, or automate notifications to users of open-source projects using GitHub developers' existing GITHUB_TOKEN in collaboration with other GitHub Enterprise Cloud features. GitHub Actions is available on hosted runners for major operating systems (OSs) (Linux, macOS, Windows) and silicon (x86, Advanced RISC Machine [ARM], graphics processing unit [GPU]). GitHub Actions can run directly on a virtual machine (VM) or inside a container. Customers can use their own VMs, in the cloud or on-premises, with self-hosted runners.

GitHub Advanced Security

GitHub Advanced Security provides features that help improve and maintain the quality and security of code. Code scanning searches for potential security vulnerabilities and coding errors in code using CodeQL or a third-party tool. Secret scanning detects secrets (e.g., keys and tokens) that have been inadvertently checked into repositories. If push protection is enabled, secret scanning also detects secrets and blocks contributors from pushing them to repositories. Dependency review and Dependabot detect vulnerable versions of dependencies and warn about the associated security vulnerabilities. Alerts from all features can be centrally reported and tracked.

System Boundaries

The scope of this report includes GitHub Enterprise Cloud service and the supporting production systems, infrastructure, software, people, procedures, and data. The following GitHub Enterprise Cloud service features are included in the scope of this report: Issues, Discussions, Pages, Projects, Docs, GitHub Advanced Security, GitHub Teams, Dependabot, Copilot, GitHub Enterprise Cloud service with data residency, and GitHub Actions, as well as pull requests, wikis, and audit logging.

Subservice Organizations

GitHub uses multiple subservice organizations in conjunction with providing its GitHub Enterprise Cloud service product. GitHub uses Sabey, Quality Technology Services (QTS), CoreSite, and Equinix to provide colocation data center services, as well as Microsoft Azure (Azure) and Amazon Web Services (AWS) to provide infrastructure hosting. These subservice organizations are excluded from the scope of this report. The expected controls for which they are responsible are found in a subsequent section titled Subservice Organizations and Complementary Subservice Organization Controls (CSOCs).

The system description in this section of the report details GitHub Enterprise Cloud service and its associated features noted above. Any other Company services are not within the scope of this report. The accompanying description includes only the policies, procedures, and control activities at GitHub and does not include the policies, procedures, and control activities at any subservice organizations (see below for further discussion of the subservice organizations).

The Components of the System Used to Provide the Services

The components that directly support the services provided to customers are described in the subsections below.

The following diagram illustrates the GitHub Enterprise Cloud service production environment:

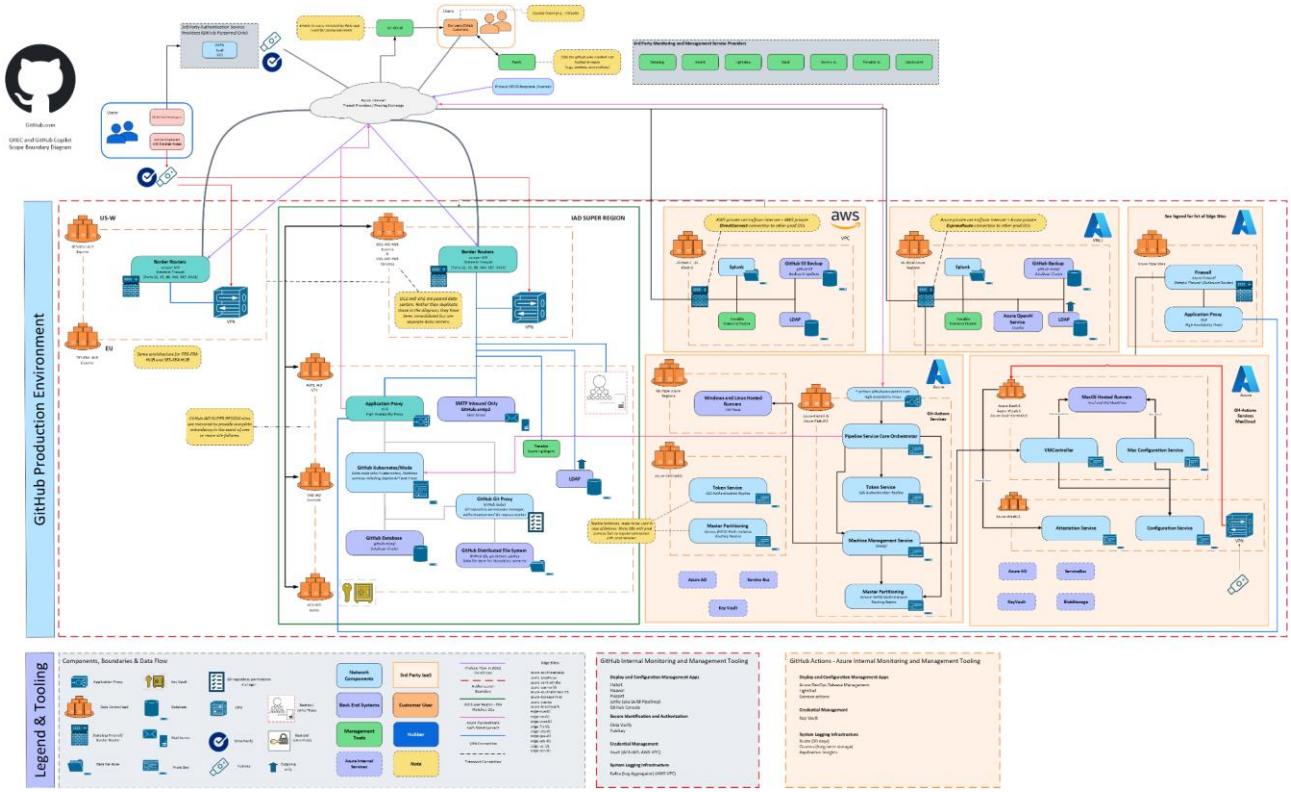

Figure 1: GitHub Enterprise Cloud and Copilot Service Production Environment

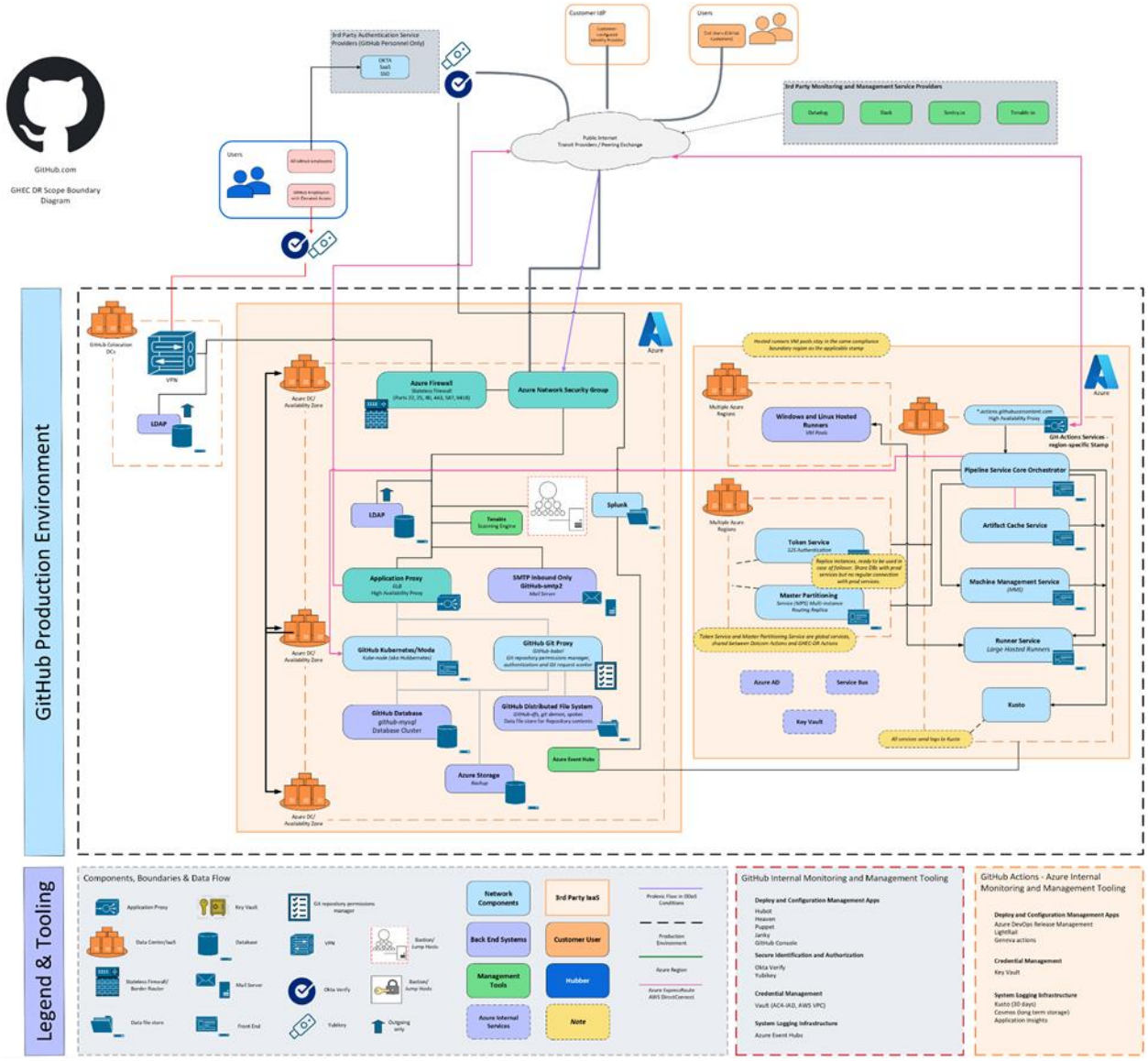

Figure 2: GitHub Enterprise Cloud Data Resident Service Production Environment

Infrastructure

GitHub utilizes Azure, AWS, and GitHub-managed colocation data centers to provide the resources to host GitHub Enterprise Cloud service. GitHub leverages the experience and resources of Azure and AWS to scale quickly and securely as necessary to meet current and future demand. However, GitHub is responsible for designing and configuring the GitHub Enterprise Cloud service architecture within Azure, AWS, and GitHub-managed colocation data centers to ensure security and resiliency requirements are met.

The in-scope hosted infrastructure also consists of multiple supporting tools, as shown in the table below, with the GitHub Enterprise Cloud service section serving as the foundation and subsequent sections building upon that foundation:

Infrastructure			
GitHub Enterprise Cloud service			
Production Tool	**Business Function**	**OS**	**Hosted Location**
Databases	Issue contents, other structured data	MySQL	Azure, AWS, and GitHub colocation data centers
Git Storage	Storing repository contents	Ubuntu	Azure, AWS, and GitHub colocation data centers
Border/Edge Routers	GitHub load balancers, application layer proxies, and firewalls are the systems that connect to the internet. These routers, application proxies, and firewalls are the first line of defense in protecting the system.	Juniper Junos, Azure VMs	Azure and GitHub colocation data centers
Bastion Hosts	Secure Shell (SSH) privileged access management	Ubuntu	Azure and GitHub colocation data centers
Production Hosts	Production compute necessary to provide the services of GitHub Enterprise Cloud service	Ubuntu	Azure and GitHub colocation data centers
Actions Hosted Runners	Production compute necessary to provide the services of GitHub Actions	Windows, macOS, Linux VMs	Azure data centers
CoreSite, Sabey, Equinix, QTS	GitHub colocation data center providers	N/A	Various locations
Azure Platform-as-a-Service (PaaS) Services	Development, deployment, and scalability	Azure VMs	Azure data centers
GitHub Enterprise Cloud service with Data Residency			
Production Tool	**Business Function**	**OS**	**Hosted Location**
Data Residency Stamps	Stamps represent geographic zone-based regions where code and repository data are stored based on the preferred region of customers.	Ubuntu, Azure VMs	Azure data centers
GitHub Copilot Business/Enterprise			
Production Tool	**Business Function**	**OS**	**Hosted Location**
GitHub Copilot Business/Enterprise API	API front-end servers are the interface to GitHub Copilot Chat clients.	Ubuntu	Azure and GitHub colocation data centers

Infrastructure			
GitHub Enterprise Cloud service			
GitHub Copilot Business/Enterprise Proxy	Service that manages connections between Code Completion plugins and AI models	Ubuntu	Azure and GitHub colocation data centers

Software

Software consists of the programs and software that support GitHub Enterprise Cloud service (OSs, middleware, and utilities). The list of software and ancillary software used to build, support, secure, maintain, and monitor include the following applications, as shown in the table below, with the GitHub Enterprise Cloud service section serving as the foundation and subsequent sections building upon that foundation:

Software	
GitHub Enterprise Cloud service and GitHub Enterprise Cloud service with Data Residency	
Production Application	**Business Function**
Azure PaaS Services	Solutions to assist in development, deployment, and scalability
Ansible	Network architecture configuration and management
Datadog	Application and infrastructure monitoring
GitHub Actions	CI/CD
GitHub Advanced Security	Dependency management, static code analysis, and secret scanning
Jamf Pro	Endpoint management and security
gPanel	In-house developed tool for data center asset inventory and management
Microsoft Defender	Endpoint security
MySQL	Account and license storage
Okta	Single sign-on (SSO) configuration management
Oomnitza	Endpoint inventory and management
Puppet	Infrastructure configuration management
RunZero	Cloud asset inventory
Slack	Chat Operations (ChatOps) and daily work communications
Splunk	Security information and event management (SIEM), logging system and intrusion detection
Tenable	Vulnerability scanning

Software	
GitHub Enterprise Cloud service and GitHub Enterprise Cloud service with Data Residency	
Production Application	**Business Function**
Terraform	Infrastructure configuration management
Ubuntu	OS baseline
Zendesk	Customer support
GitHub Copilot Business/Enterprise	
Production Application	**Business Function**
Azure OpenAI (Azure) Anthropic (AWS) Gemini Google Cloud Platform (GCP)	AI models (GPT-4o, GPT-4.1, o1, 03, o3-mini, o4-mini) AI models (Claude 3.5 Sonnet, Claude 3.7 Sonnet) AI models (2.0 Flash)

People

GitHub develops, manages, and secures GitHub Enterprise Cloud service via separate departments. The responsibilities of these departments are defined in the following table:

People	
Group/Role Name	**Function**
Customer Support	Responsible for providing technical and account-related support to GitHub Enterprise Cloud service customers and for resolving customer issues via email, chat, social media, and phone from developers and customer entities around the globe.
Engineering	Responsible for working with the Product Management team to plan and coordinate releases, and is accountable for building, testing, and deploying GitHub Enterprise Cloud service code and feature changes. Responsible for maintaining service availability, including performance and scale monitoring and reporting, incident command, and on-call readiness for any production issues. Responsible for configuration management; building, testing, and deploying software relevant to the operation and management of production assets; patching and remediating vulnerabilities reported by the Security team; and data center operations management. Responsible for managing Git and database storage backups and restores.
Legal	Responsible for negotiating contractual obligations with third parties and technology partners/suppliers, legal terms and conditions, and ensuring compliance with internal contractual standards.
People Operations	Responsible for talent acquisition, diversity and inclusion, learning and development, and employee engagement on everything from benefits and perks to career development and growth.
Privacy	Responsible for determining which privacy laws and regulations apply to GitHub and determining the best way to comply with them, ultimately ensuring GitHub can offer its products to every developer anywhere in the world.

People	
Group/Role Name	**Function**
Product Management	Responsible for understanding customer requirements; collecting, defining, and clarifying feature requests and development efforts; and managing feature rollouts and related customer communication efforts.
Security	Responsible for ensuring the security of GitHub products. Security consists of multiple teams with specific missions: Computer Security Incident Response Team (CSIRT)); Product Security Incident Response Team (PSIRT); Security Lab; Security Operations; Secure Access Engineering; Security Telemetry; Vulnerability Management; Cloud and Enterprise Security, Risk, and Governance and Trust. These teams manage security incident detection and response, monitoring, vulnerability scanning, network and application layer penetration testing, security architecture, security engineering and operations, access management, endpoint asset management, and risk and compliance oversight.
Senior Leadership	Responsible for the overall governance of GitHub. This group includes the Chief Executive Officer (CEO), Chief Financial Officer (CFO), Chief Revenue Officer (CRO), Chief Technology Officer (CTO), Vice President - Chief Information Security Officer (CISO), Chief People Officer (CPO), Head of Design, Chief Operating Officer (COO), Chief of Staff, Vice President, Senior Vice President of Engineering, Chief Product Officer, and Chief Legal Officer.

Policies, Standards, and Procedures

GitHub maintains policies, standards, and procedures necessary to securely operate GitHub Enterprise Cloud service. Policies and standards are centrally managed in The Hub, GitHub's centralized internal communication platform. The Hub is backed by a repository, which is used to implement annual reviews and control changes to policies and standards. Once a change has been approved by the owner of the policy or standard, it is automatically updated on The Hub. Procedures are developed and documented within the GitHub repositories maintained by every team to provide end-user documentation and guidance on the multitude of operational functions performed daily by GitHub security and product engineers, developers, administrators, and support personnel. These procedures are drafted in alignment with the overall policies and standards and are updated as necessary to reflect changes in the business.

GitHub policies and standards establish controls to enable security, efficiency, availability, and quality of service. The GitHub Information Security Management System (ISMS) Policy and related policies define information security practices, roles, and responsibilities. The ISMS outlines the security roles and responsibilities for the organization and expectations for employees, contractors, and third parties utilizing GitHub systems or data.

This overarching security policy is supported by several dependent security policies, standards, and procedures applicable to the operation and management of security across the organization. Security-related policies, standards, and procedures are documented and made available to individuals responsible for their implementation and compliance.

Below is the current inventory of security and audit-related policies and standards that inform procedures operating in support of the GitHub ISMS Policy objectives:

Procedures	
Policy	**Associated Standards and Procedures**
GitHub ISMS Policy	• ISMS Internal Audit Procedure • ISMS Corrective Action Procedure
GitHub ISMS Scope Policy	No associated standards or procedures
GitHub ISMS Statement of Applicability (SOA)	• ChatOps Command Security and Risk Standard • Controls Monitoring Standard • Controls Monitoring Standard Operating Procedure (SOP) • Data Classification Standard • Domain Management Standard • Endpoint Security Standard • Enterprise Administration Standard • External File Sharing Standard • Git Systems Server Site Failure Plan • High-Risk Application Access Standard • Organization Administration Standard • Production VPN Access Standard • Repository Security Baseline Configuration Standard • Reviewing Pull Requests • Server Operating System Standard
Corporate Data Retention Policy	• Audit Video Retention Standard • Corporate Data Retention Standard • Product Telemetry Data Retention Standard • Slack Retention Standard
Contractor Termination Policy	No associated standards or procedures
Full-Time Employee Termination Policy	No associated standards or procedures
Identity and Access Management (IAM) Policy	IAM Standard • IAM Onboarding SOP • IAM Entitlements SOP • IAM Privileged Systems and Elevated Access SOP • Granting Slack Access to Contractors and Consultants SOP • IAM Non-Human Accounts in Okta • IAM Offboarding SOP • IAM On-Leave SOP
Physical and Environmental Protection Policy	Production Datacenter Standard • Datacenter Physical Access SOP • Production Media Destruction SOP • Datacenter Access Compliance Guidelines
Privacy Statement	No associated standards or procedures

Procedures	
Policy	**Associated Standards and Procedures**
Private Information Removal Policy – External Customer Facing Policy	No associated standards or procedures
Secure Coding Policy	Secure Coding Standard • Secure Coding – Dotcom • Secure Coding – General Guidance • Security Requirements for New Applications
Security Awareness and Privacy Training Policy	Security Awareness Training Standard
Security Event Logging and Monitoring Policy	• Security Event Logging Standard • Security Event Logging SOP • Security Event Monitoring and Alerting Standard
Security Incident Response and Data Breach Notification Policy	• Data Breach Notification Standard • Security Incident Response Standard • Security Incident Response Procedure • Data Breach Notification Procedure • Security Concern Reporting Procedure
Security Policy Exception Policy	No associated standards or procedures
Security Risk Management Policy	• Security Governance Risk Compliance Cybersecurity (GRCC) – Vendor Risk Assessment Process • Security Risk Reporting – Standard Operating Procedure • Centralized Key Management Procedure • GitHub Attestation CA Certification Practice Standard • Secure AI Product Development Standard • Cloud Standard
System and Services Acquisition Policy	• Vendor Security Standard • Purchasing Workflow • Vendor Security Reviews SOP • Procurement Workflow • Vendor Offboarding Checklist • Decommissioning a GitHub-Owned App • Vendor Offboarding SOP • Encryption Standard
Vulnerability Management Policy	• Container Hardening Standard • Exception Handling Process • Vulnerability Management Process • Docker Baseline Security Checklist • Database Hardening Standard • OS Hardening Standard • Patch Management Standard • FedRAMP Vulnerability Reporting Standard • FedRAMP Annual Vulnerability Exception Review • FedRAMP Monthly Vulnerability Management Reporting Procedure

Procedures	
Policy	**Associated Standards and Procedures**
Background Checks Policy	No associated standards or procedures
IT Asset Management Policy	No associated standards or procedures
Network Policy	No associated standards or procedures
Resilience Program Policy	Resiliency Standard
Security Document Management Policy	Security Document Management Standard

Data

Data refers to transaction streams, files, data stores, tables, and output used or processed by GitHub. GitHub uses repository data to connect users to relevant tools, people, projects, and information. Repositories are categorized as public, private, or open source. Public repositories can be viewed by anyone, including people who are not GitHub users. Private repositories are only visible to the repository owner and collaborators that the owner specified. GitHub aggregates metadata and parses content patterns to deliver generalized insights within the product. It uses data from public repositories and uses metadata and aggregate data from private repositories when a repository's owner has chosen to share the data with GitHub through an opt-in.

If a private repository is opted in for data use to take advantage of any of the capabilities of the security and analysis features, then GitHub performs a read-only analysis of that specific private repository's Git contents. If a private repository is not opted in for data use, its private data, source code, or trade secrets are classified internally as restricted, and they are maintained as confidential and private consistent with GitHub’s Terms of Service.

Customer data is managed, processed, and stored in accordance with relevant data protection and other regulations and with specific requirements formally established in client contracts.

GitHub has deployed secure methods and protocols for transmission of confidential or sensitive information over public networks.

The following table details the types of data contained in the production application for GitHub Enterprise Cloud service:

Data		
GitHub Enterprise Cloud service and GitHub Enterprise Cloud service with Data Residency		
Production Application	**Description**	**Data Store**
GitHub Enterprise Cloud service	GitHub stores data related to customer repositories, including but not limited to code, file content and diffs of content, issues, commits, comments, uploaded files, build logs for private repositories, pages, and discussions.	Git Storage, MySQL
GitHub Advanced Security	GitHub stores data related to Advanced Security features, including but not limited to Dependabot alerts, secret scanning results, and CodeQL results.	MySQL

Data		
GitHub Enterprise Cloud service and GitHub Enterprise Cloud service with Data Residency		
GitHub Actions	GitHub stores data related to Actions features, including but not limited to customer-provided runner images, Actions configurations, and runner logs and output.	MySQL
Telemetry Data	Telemetry data is gathered via Datadog and Splunk by GitHub to facilitate the logging and monitoring of GitHub Enterprise Cloud services. GitHub logs information about customers and their users, including Internet Protocol (IP) addresses. These records may be used to assist in detecting security violations, performance problems, and flaws in applications.	SIEM services
Authentication/License Data	GitHub manages authentication services and associated data that authenticate customer users (or manage the delegation of authentication to customer identity providers).	MySQL
GitHub Copilot Business/Enterprise		
Production Application	**Description**	**Data Store**
Prompts	“Prompt” refers to the collection of code and supporting contextual information that GitHub Copilot Business/Enterprise sends to GitHub to generate Suggestions. This also includes the content that users submit through a chat interface.	Not stored when using GitHub Copilot Business Chat and Code Completion in the IDE. Stored in CosmosDB when using other Copilot features.
Suggestions	“Suggestions” refers to the code, functions, and other output returned to a user by GitHub Copilot Business/Enterprise.	Not stored when using GitHub Copilot Business Chat and Code Completion in the IDE. Stored in CosmosDB when using other Copilot features.
User Engagement Data	User engagement data is usage information about events generated when interacting with a code editor. These events include user edit actions (e.g., whether a suggestion was accepted or dismissed, but not the content of the suggestion), error messages, and general usage data to identify user metrics such as latency and feature engagement. This information may include personal data, such as pseudonymous identifiers.	GitHub Copilot Telemetry Service
Pull Request Summaries (Enterprise)	Pull request summaries allow Copilot Enterprise customers to create a summary of the changes that were made in a pull request, which files they impact, and what a reviewer should focus on when they conduct their review.	Not stored until the user accepts the Copilot suggested content. Once accepted, it is stored in MySQL as part of the standard pull request storage process.

Significant Events and Conditions

Customers are responsible for submitting incident tickets through the customer portal when an incident or system disruption is identified. Customer requests are recorded and tracked with an internal ticketing system through resolution. The ticketing system is utilized to document, prioritize, escalate, and resolve problems affecting contracted services.

Relevant Aspects of the Control Environment, Information and Communication, Risk Assessment and Mitigation, Monitoring, and Control Activities

The five interrelated components of internal control at GitHub include:

- *Control Environment* – sets the tone of an organization, influencing the control consciousness of its people. It is the foundation for all other components of internal control, providing discipline and structure.
- *Information and Communication* – are systems, both automated and manual, that support the identification, capture, and exchange of information in a form and time frame that enable people to carry out their responsibilities.
- *Risk Assessment* – is the entity's identification and analysis of relevant risks to the achievement of its objectives, forming a basis for determining how the risks can be managed.
- *Monitoring* – is a process that assesses the quality of internal control performance over time.
- *Control Activities* – are the policies and procedures that help make sure that management's directives are carried out.

GitHub's internal control components include controls that may have a pervasive effect on the organization, an effect on specific processes, account balances, disclosures, classes of transactions or applications, or both. Some of the components of internal control include controls that have more of an effect at the entity level, while other components include controls that are primarily related to specific processes or applications. When assessing internal control, GitHub considers the interrelationships among the five components.

Control Environment

Integrity and Ethical Values

The internal control environment reflects the overall attitude, awareness, and actions of executive management and other stakeholders concerning the importance of controls and the emphasis given to controls in GitHub's policies, procedures, methods, and organizational structure.

Management is responsible for directing and controlling operations and for establishing, communicating, and monitoring policies and procedures. Maintaining sound internal controls and establishing the integrity and ethical values of personnel is a critical management function.

During the onboarding process, new employees and contractors complete security and privacy awareness training and review and acknowledge the employee guide to company policies and practices, the Handbook for Hubbers. The handbook includes the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics. An annual Standard of Business Conduct training is mandatory for all employees and contractors. This training covers key policies, describes how to report

issues, and emphasizes the importance of ethical behavior. Each employee and contractor must attest to the completion of the training. Any ethical issue reported is investigated, and appropriate action is taken, up to the termination of employment.

Board of Directors

As a wholly owned Microsoft subsidiary, GitHub management directs the strategy and operations of the business but is accountable to Microsoft's management and reporting structures, including the Microsoft Board of Directors. GitHub does not have a separate Board of Directors. The GitHub CEO, Thomas Dohmke, meets with leaders within Microsoft regularly (bimonthly, monthly, and quarterly), and GitHub is ultimately reported on to Microsoft's Board of Directors through processes within Microsoft. GitHub relies on these Microsoft processes.

Organizational Structure

GitHub's organizational structure provides the framework within which its activities for achieving company-wide objectives and key results are defined, planned, sponsored, executed, controlled, and monitored. Management believes that establishing a relevant organizational structure includes considering key areas of authority and responsibility and lines of reporting. Senior Leadership sets company-wide objectives and key results and reports on and reviews progress toward meeting those objectives semiannually.

GitHub has established appropriate lines of reporting considering the nature, size, and culture of GitHub. People Operations maintains an organizational chart that outlines security responsibilities across GitHub. The organizational chart is available for employees on GitHub's intranet and the internal human resource information system (HRIS) and is updated through automation. Management continues to evaluate its organizational structure and makes changes as necessary. Hubber responsibilities are communicated through documented job descriptions, which are maintained by GitHub.

GitHub's organizational structure is designed to meet GitHub's control objectives. GitHub's Senior Leadership team, specifically those reporting to the CEO, provide direction and oversight, and includes members who are independent from control operations. In addition, management is responsible for establishing, communicating, and monitoring policies and procedures, as well as aligning operations with leadership's defined objectives and key results.

When leadership changes occur within the organization, the Security team is notified when there may be a compliance impact. New incoming leadership is announced to the organization on GitHub intranet and in all-hands meetings.

The Security team, CTO and Vice President CISO, are responsible for monitoring and enhancing the GitHub's overall security posture. This function includes managing security risks and threats, educating employees on security-related practices, building security awareness, responding to security incidents, and performing internal security audits and security reviews. Security leadership considers out-of-band changes based on newly identified risk findings or service changes and addresses such changes as deemed appropriate.

Security leadership is also accountable for planning and staffing appropriately to address risk remediation and mitigation as identified in prescribed risk monitoring activities. Security leadership reviews these components as part of an annual headcount and budgeting process.

Management’s Philosophy and Operating Style

Management across GitHub is accountable for ensuring necessary policies, standards, and SOPs aid in assessing and addressing operational risks. These owners review and update these policy-related documents at least annually, to reflect changes and help ensure completeness and accuracy, based upon the annual risk assessment, based upon strategic business initiatives driven by leadership, and other events that dictate changes. Security leadership reviews and approves the materials and any changes annually.

The Security team administers security and privacy awareness training to personnel during their new hire onboarding, and annually thereafter. The Security team follows up with employees who are delinquent in completing the training until these employees are compliant. Moreover, periodic security awareness notifications are sent to employees as needed, highlighting new controls or warning them of known threats.

Authority and Responsibility

All layers of GitHub management are accountable for managing objectives and key results and established policies and standards. Managers are expected to engage with the Security and Legal organizations in appropriate work, decision making, and response to reduce risk to GitHub, customers, users, and employee data. They are responsible for day-to-day oversight of employees and nonemployees regarding their obligations to ensure compliance with security, privacy, and risk management requirements.

Employees are accountable for understanding their individual responsibilities as outlined in the Handbook for Hubbers and are individually responsible for designing tools and features in a secure manner, as well as ensuring issues and findings that they identify that impact security are raised to the appropriate management contact or directly to the Security organization. Individuals are also accountable for executing their work with a security-first mindset, ensuring compliance with the organization's security policies, standards, and procedures, regardless of level and role.

Non-employees, including contingent workers, contractors, and vendors, are accountable for understanding and upholding the contractual obligations to GitHub for data protection and for complying with management oversight provided to ensure understanding of those obligations.

The CTO is responsible for the overall security posture, programs, and capabilities within GitHub. The CTO works with members of GitHub's leadership team and management, Product Management, Engineering, business and system owners, and users to develop and implement prudent security services, policies, procedures, and controls, subject to the approval of leadership at GitHub.

Specific responsibilities of GitHub Security leadership include but are not limited to:

- Ensuring security policies, standards, and procedures are in place and understood by Hubbers
- Providing basic security direction and support for all systems and users
- Advising software and product design and engineering, corporate systems development, and business owners in the implementation of security controls from the point of system design, through testing, implementation, and decommissioning
- Educating employees at onboarding about security controls and processes relevant to GitHub
- Providing ongoing employee security skills and awareness education
- Performing and facilitating product security audits
- Reporting regularly to GitHub leadership on GitHub's status regarding information security

Human Resources

GitHub People Operations is responsible for the GitHub employee lifecycle. GitHub evaluates candidates' abilities in the interview process against established job descriptions. Fit to the role is scored, tracked, and approved by the hiring manager in GitHub's recruitment management system.

To be employed by GitHub, candidates must successfully complete a background check as permissible by local laws and regulations. The background check is initiated after the candidate signs the offer letter. Employees and contractors are not allowed to onboard until the background check is cleared. The Talent Coordinator is notified once a candidate clears, which is then relayed to the hiring manager. In instances where negative or incomplete information is obtained, the Vice President of Global Talent Acquisition, or a delegate, assesses, in consultation with Legal, the potential risk and liabilities related to the job's requirements and makes a final decision on the hire. Before any adverse action is taken based on a background check, GitHub provides the applicant with a notice that includes an opportunity to respond or clarify any discrepancies. If GitHub does not hire a candidate based on the results of a background check, GitHub provides the candidate with an adverse-action notice and informs them of their rights to see the information reported and to correct inaccurate information.

New employees undergo a company orientation session, at which time they are provisioned with a Company-issued laptop and their GitHub organization and relevant system credentials as commensurate with their new role. During orientation, new employees are introduced to the roles of Security, Risk, Governance and Trust, and data protection in GitHub. In addition to attending orientation, new employees and contractors are required to complete security and privacy awareness training within 45 days of joining GitHub.

GitHub requires that management and peers evaluate and provide feedback to employees annually in accordance with a process led by People Operations.

People Operations is responsible for the processes associated with Hubbers leaving GitHub. When Hubbers leave GitHub, People Operations notifies the appropriate parties within GitHub to ensure that property is collected and accounts are terminated within 24 hours of their departure.

Information and Communication

To help align GitHub business strategies and goals with operating performance, management is committed to maintaining effective communication with employees and customers.

Internal Communications

GitHub has published policies and procedures, both included in the Handbook for Hubbers and published separately, outlining the responsibility of employees to report security and operational failures, incidents, system problems, and complaints. The document owners and Security leadership review and approve security policies and procedures annually. Significant changes to policies result in communication to personnel regarding policy updates. Policies and standards are available on The Hub, GitHub's centralized internal communications platform.

Every Hubber, depending on their role, is responsible for designing and executing work and services in a secure manner in alignment with company standards and policies and for reporting issues and findings that impact security up their management chain or directly to the Security team. GitHub's employee handbook contains ethics expectations for Hubbers, including compliance with Microsoft's Standards of Business Conduct, instructions for reporting ethics and integrity concerns to Microsoft (including anonymous channels), and details of GitHub's policies against retaliation for reporting concerns.

External Communications

GitHub communicates the description of GitHub Enterprise Cloud service systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints using the external website (https://github.blog). The blog maintains a changelog, which is a chronological list of information on customer-facing feature changes, bug fixes, or security notifications made available to end users. GitHub's changelog is live on the blog site (https://github.blog/changelog) and available for RSS feed subscription. It is also accessible via GitHub's changelog X (formerly known as Twitter) account (@GHchangelog).

GitHub Enterprise Cloud service users have access to support resources at GitHub Support (https://support.github.com) and GitHub Docs (https://docs.github.com).

Customer commitments and responsibilities are communicated through GitHub's Terms of Service (https://docs.github.com/en/site-policy/github-terms/github-terms-of-service) as well as in contracts. The Terms of Service includes GitHub's Acceptable Use Policy, which provides the basic rules customers are required to follow as members of the community on the GitHub Enterprise Cloud service product.

The user community can communicate directly with GitHub. GitHub Customer Support receives reports of security issues and many other end-user concerns and questions via email or through the "Contact Us" web form. A ticketing system is used to document and track these issues to resolution.

Risk Assessment and Mitigation

GitHub recognizes that risk management is a critical component of its operations and contributes to ensuring customer data is properly protected. GitHub incorporates risk management throughout its business processes and across the organization. The foundation of this process is management's knowledge of its operations, its close working relationship with its customers and vendors, and its understanding of the space in which it operates.

The Risk team is embedded within the larger Security team. This team monitors GitHub's internal controls and conducts risk monitoring in accordance with relevant regulatory and contractual compliance requirements. This responsibility includes managing the design, implementation, and monitoring of the GitHub Enterprise Cloud service control environment, as well as assessing, monitoring, and mitigating security, fraud, and compliance risks.

Vendor Management

To initiate a vendor relationship, the Legal and Procurement teams negotiate and manage the vendor contract clauses and additional data protection agreements with vendors who process or store GitHub data, customer data, or employee data, as well as systems that connect to GitHub systems.

The Risk team manages the vendor security risk assessment process. GitHub maintains operational processes to assess security risk considerations related to vendors. These vendors are required to undergo an initial security risk assessment prior to contracting with GitHub. Those vendors who do not meet GitHub's baseline security requirements in alignment with their defined business use case do not move forward for procurement.

The Risk team maintains an inventory of approved vendors and reviews vendor security risk assessments every two years or upon expansion or changes to the contracted service offering. Vendors deemed high risk, such as data center providers or other vendors storing or processing data in scope for GitHub's regulatory or contractual requirements, undergo reassessment annually.

Risk Identification and Treatment

GitHub has defined risk management processes to identify and manage risks that may affect the system's security. At least annually, the Risk team performs a risk assessment to identify, track, and treat technical risks related to the product and reports risks to GitHub leadership. Risks identified are documented in a GitHub repository where the risks and mitigation actions are tracked to resolution.

Risk Reporting

Summary reporting on security risk areas is included in annual leadership reporting as part of the annual security risk assessment. GitHub's Senior Leadership team, as well as extended leadership teams across Security, Engineering, Information Technology (IT), Business Systems, Legal, and Privacy, review this annual risk report.

Vendor Inventory

A vendor inventory is generated each quarter as part of regulatory reporting for other compliance requirements. The Risk team generates this inventory using automated tools.

Vulnerability Management

The GitHub Vulnerability Management Program monitors and interrogates public-facing and internal infrastructure to identify systems that are vulnerable to known exploits, are configured in ways that unnecessarily increase risk of compromise, or have software installed that is known to be insecure. The program develops and maintains scanning coverage across all hosting platforms, providers, and networks. Vulnerability scanning is executed monthly, with findings prioritized for remediation based on risk, technology dependencies, and exposure.

GitHub has a published internal standard based on the Common Vulnerability Scoring System (CVSS) levels for vulnerability management (critical, high, medium, and low) as reported by GitHub's vulnerability scanning and reporting system. GitHub's Vulnerability Management team uses the CVSS score for initial vulnerability assessment and then conducts an impact analysis that takes into consideration the specifics of GitHub's infrastructure. When a credible and actionable vulnerability is identified, the Vulnerability Management team provides the system owner with a description of the perceived impact of the vulnerability and the required timetable for resolution/mitigation. Vulnerabilities that cannot be remediated within the required service-level agreement (SLA) are handled via the documented exception process.

GitHub Product Security Engineering, Vulnerability Management Engineering, and Infrastructure teams triage vulnerabilities from vendors and internal and external scanning, and patch those vulnerabilities based on risk and exposure. GitHub prefers to run "known good" software that has been tested and operated in production. Preference is given to running the most stable release of software possible. Patches are not generally applied for the objective of running the latest or "bleeding-edge" version of any package.

Patching occurs to address security fixes, bug fixes, and performance issues and to accommodate new features. Patches are applied through a combination of automated and manual processes. Linux servers automatically install most security patches via an unattended upgrade process that is orchestrated by GitHub's configuration management system.

In addition to the production network vulnerability assessments, the GitHub Product Security Engineering team provides application security services to the Engineering teams. These services include secure architecture and code review, the development and maintenance of internal automated security testing, and secure code training.

Bug Bounty

The Bug Bounty team manages the GitHub Security Bug Bounty program, hosted on HackerOne. Members of the GitHub and security research community are encouraged to submit vulnerabilities through the program. The on-call security resource monitors the submissions and triages accordingly. If a bug is deemed to be legitimate, Security informs the relevant engineers, and the bug is tracked in GitHub Issues to resolution. The Bug Bounty team issues bounty rewards for significant finds that lead to security improvements to the platform.

Penetration Testing

GitHub engages with a third-party security vendor to execute penetration and application security testing annually. The scope of this testing varies from year to year to focus on areas assessed as presenting significant risk, such as new features or services. Results are triaged and the risks reported are assessed against internal environmental considerations. Where remediation is required, solutions are identified and assigned to the relevant engineering teams for resolution with appropriate prioritization. Upon remediation, the fixes performed are shared with the contracted vendor to confirm successful remediation. A customer-facing report of the annual security testing is available to clients under mutual nondisclosure agreements.

Monitoring

The Governance and Trust team at GitHub is responsible for monitoring the internal control environment for each of the compliance frameworks adopted by GitHub.

Internal Control Reviews

The Governance and Trust team conducts internal control assessments annually to assess the effectiveness of the internal control environment. Control assessment results are formally documented and retained. Issues are identified and escalated to the relevant internal teams to resolve control design or effectiveness issues, and the status of the operating effectiveness of controls, identified gaps, and remediation efforts are reported quarterly to the Senior Leadership team for awareness. Additionally, an internal International Organization for Standardization (ISO) audit is conducted each year; because GitHub does not have its own independent audit function, this audit is performed by a certified ISO audit firm, and results are reported to management in accordance with audit processes.

Microsoft maintains an independent internal audit function, Microsoft Internal Audit (MSIA), that assesses GitHub as required by the Audit Committee of the Microsoft Board of Directors. Results of MSIA engagements are reported to GitHub management, as well as the Audit Committee of the Microsoft Board of Directors, in accordance with audit processes.

Control Activities

Control activities have been established to help ensure key processes operate as intended. These activities are integrated into the policies, standards, and procedures outlined in the Procedures section above.

Logical Access

GitHub Enterprise Cloud Service Employee Access

GitHub has implemented access protection measures to only allow authenticated and authorized users access to the portions of GitHub's instance of GitHub Enterprise Cloud service that are not explicitly public. Access to GitHub's instance of GitHub Enterprise Cloud service requires a valid and unique account ID, password, and two-factor authentication (2FA). GitHub integrates with a third-party, SSO provider that enforces 2FA using phishing-resistant authenticators.

Access to internal systems is restricted through unique account IDs (handles), passwords, and 2FA. To access these environments, GitHub users leverage the same GitHub handles as GitHub Enterprise Cloud service. Employees are prohibited by policy from using shared accounts or credentials when accessing GitHub internal systems. Exceptions to the policy are documented in the Exceptions Policy. Employee access to GitHub Enterprise Cloud service production systems is restricted to authorized personnel with demonstrated need and isolated from the Internet by virtual private network (VPN), bastion hosts, and/or a Security Assertion Markup Language (SAML) solution enforcing Hypertext Transfer Protocol Secure (HTTPS) reverse proxy. GitHub uses SSH to authenticate to back-end production resources through a bastion host. VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.

Employee access to production infrastructure must be approved by designated personnel before access is granted, and that approval must be renewed during periodic access reviews. Server and database administrative privileges are restricted to the respective system owners. Individual users, teams, and managers request and manage access via the GitHub Entitlements system. The Secure Access Engineering team has the ultimate responsibility for maintaining the Entitlements system, but the access grants and review flow are the responsibility of the users requesting access and their manager.

GitHub Enterprise Cloud Customer Access

Authentication to GitHub Enterprise Cloud service requires a unique account and password. Customers can also enable 2FA for their organization through GitHub Enterprise Cloud service settings or integrate with their identity provider via SSO. Audit logging is also available to allow organization administrators to quickly review actions performed by users. This log includes details such as who performed the action, what the action was, and when it was performed. Where a repository is configured as private, GitHub Enterprise Cloud service is designed to limit access to data and settings to the user who created the repository or users who have been explicitly granted access.

GitHub file servers are obfuscated from front-end applications, where each Git operation, such as read, commit, or push in a repository, is directed through GitHub's authentication pathway using service handlers. Requests terminate at the Git proxy servers, where GitAuth, the service handler endpoint, is executed to perform authentication and repository permission checks.

GitAuth checks whether the requested repository is active or disabled and is public or private, and checks the user permissions and authentication information (e.g., anonymous user for public only access, username and password, OAuth token, or SSH keys). If the service provides a successful response for the authentication and permission check, GitAuth returns the routing information for access to the file server host where the repository can be found. If the response is denied, the request returns a 404 error.

Network Security

GitHub's network security is enforced through several process and configuration controls to protect against unauthorized access and help ensure security of data in transit. Both stateless and stateful network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary.

Access to production systems is brokered through security gateway systems at the production network perimeter. Three technologies provide access: a VPN, a bastion host, and a SAML-enforcing HTTPS reverse proxy.

Each of these remote access mechanisms requires multiple factors for authentication and employs strong encryption in transit. Employees are provided access to production systems based on their role within the organization. Access is granted through the GitHub Entitlements system, which updates the internal Lightweight Directory Access Protocol (LDAP) store with the correct authorization rights for the user.

Access to machine accounts is limited and activity is logged and audited. Access to production systems requires the use of an authorized SSH key with 2FA enabled.

GitHub personnel can only elevate privileges for their account and run commands with sudo if they are members of the correct Entitlements group, which is strictly controlled and reviewed. Any of these actions, including elevating to root, require using the sudo command and are logged in the SIEM tool. Alerting is configured to detect unusual or unauthorized activity, and, when triggered, a response is executed in accordance with the standard procedures defined within the PSIRT's monitoring, detection, and response program. As needed, based on the frequency and type of activity detected, the PSIRT identifies process improvements to reduce human interactions directly with production servers.

Encryption

GitHub Enterprise Cloud service traffic is encrypted in transit over the public internet. Transmissions via Transport Layer Security (TLS) require a minimum of 2048-bit certificate keys, and GitHub uses TLS 1.2 and above to encrypt data during transit.

Internal User Provisioning

New employee permissions are added during the onboarding process based on predefined permissions granted to individuals based on their team and their role within that team. The user's manager and the Security Operations team review and approve any additional privileges granted outside of those provisioned as part of their role. GitHub's internal systems are configured to automatically provision non-role-based Entitlements only after the user's manager and Security Operations team review and approve the access request.

During onboarding, GitHub IT guides employees through configuring their account setup, 2FA, and secondary mobile device authentication tools for GitHub's SSO provider and the GitHub organization. Users are locked out of internal resources until remediated if they are found to not have the required security settings enabled in accordance with requirements.

The Security Operations team is responsible for operating periodic access tooling for sensitive systems. The Security Operations team identifies systems and elevated access that pose significant risk to the organization. Specifically, these systems include production, security management tools, user access tools, and vulnerability management tools. Managers review, approve, and accept the risk for their user's logical access to critical production and access gateway systems semiannually. The reviews allow the manager to provide attestation that the level of access is appropriate and required for the job. User access levels remaining after changes from the review are authorized by management or are removed if approval is not granted within seven days.

The Infrastructure team reviews physical data center access annually. Accounts belonging to unapproved individuals are removed.

CSIRT builds automated monitors of user access events to identify anomalous user access activity. The PSIRT investigates detections, and if PSIRT concludes that the activity is potentially malicious, the account is locked to reduce risk of unauthorized access and incident response procedures are initiated.

Deprovisioning occurs within 24 hours of a user's termination. An offboarding notification to Security Operations is pulled from Workday, where People Operations updates employee termination dates, for both planned and unplanned exits. The Security Operations team monitors the automation system for errors and can manually trigger offboarding when required.

The timeframe for completing access removals for corporate and production network resources and physical access is within 24 hours of notification of the termination. High-risk access is deprovisioned first, sometimes manually at the time of termination, to ensure a terminated employee can no longer access GitHub's sensitive internal resources.

System Operations

System Monitoring

CSIRT actively employs and monitors a wide variety of data logging and telemetry sources across the organization. Production systems and corporate infrastructure transmitting, processing, or storing GitHub data are configured to generate and transmit security event logs. These include:

- Enterprise – system monitoring, audit logs, application logs, and network telemetry
- Corporate – endpoint system, network telemetry, cloud service, and application logs

These logs are actively monitored for a variety of known security issues, nefarious activity, malicious indicators, privileged actions, unauthorized access, and other anomalous activity. GitHub maintains a comprehensive security detection framework that governs how threat and anomaly detection is performed.

Detection alerts are generated by a variety of systems, including log query tools, network or endpoint monitoring systems, or orchestration tooling. Each detection is assigned a severity and initial response SLA based on the risk represented by the events it is designed to detect and surface. SLAs are as follows:

Severity	SLA	Intended Response	Example
Critical	30 minutes, 24/7	Immediate human response	Confirmed successful phishing attempt against Hubber
High	24 hours	Human response within one day	High-confidence indicator of compromise (IOC) match or behavioral detection
Medium	1–7 days	Automated response or human response within a week	Low-confidence IOC match
Low	N/A	Automated or contextual response only	Uptick in password changes from a single IP

Each alert generated by detection monitoring is assigned to the relevant security team or accountable individual at the time of creation, based on the severity assigned. The assigned team or individual records any actions taken or postmortem conclusions in the relevant security repository issue for that event and resolves the issue when complete.

Incident Response

CSIRT and PSIRT triage, investigate, and respond to a variety of security events reported by internal and external sources, including:

- Automated log monitoring and alerting
- Suspected internal security compromises
- Critical GitHub Enterprise Cloud service vulnerabilities or bugs that may expose user or customer data

GitHub maintains documented incident response procedures. These procedures are triggered once an event is detected and reported. GitHub security personnel are informed of security events, whether detected by systems or reported by humans, internal or external, through formally documented procedures, with alerts surfaced to responders based on their severity.

The first step in any incident response process is to determine what constitutes an incident. Incidents are then classified by severity using "sev" definitions, with lower numbered severities being more urgent. Operational issues are classified at one of these severity levels. Anything above a Sev-2 is automatically considered a "major incident" and gets a more intensive response than a normal incident.

If GitHub is unsure about an incident's level (e.g., not sure if it is Sev-2 or Sev-1), the incident is treated as the higher level. During an incident is not the time to discuss or litigate severities.

Incident severity levels are described in the following table:

Incident Severity Ranking		
Severity Level	**Description**	**Action Taken**
Sev-0 (Disaster)	**Critical issue that warrants a public statement and liaison with executive teams.** • One or more systems in a critical state globally due to a physical or geopolitical situation (Data Center (DC) and network capacity, workspace, infrastructure) or a security issue that results in GitHub not being delivered at a level that is intended. • Irrecoverable data loss or data corruption due to platform defect or security incident. • Customer data-exposing security vulnerability has come to GitHub's attention.	Page an Incident Manager in Slack or escalate within the current Incident Responder team.
Sev-1 (Single/multi-service/region impact)	**Issue that warrants a public notification.** • A system is in a degraded or disrupted state and is actively impacting a subset of customers. • Impact is significant: either widespread or sustained (more than four hours). • Functionality has been impaired, potentially breaking SLA.	Service owner team to investigate and mitigate.
Sev-2	**Issue that may have a customer impact if not mitigated in the near future.** A system showing a trend towards an unhealthy state such as disk space usage reaching a warning level.	Service owner team to investigate and mitigate.
Sev-3	**Operational issue** A single transient failure of a cron job, which is due to run again.	Service owner team to investigate and mitigate.

Initial reports are defined as "leads" until appropriately triaged by a member of the relevant PSIRT team. Once either CSIRT or PSIRT verifies a lead, an escalation procedure is executed to engage appropriate resources to help ensure any potential risk of breach or privacy concerns are addressed in accordance with established protocols. Cross-functional accountabilities are documented for reference, as escalation and remediation engagement vary depending on the source, scope of impact, and severity of the incident.

Roles for incident response and handling are defined and assigned, and a multi-stage process is followed until the risk has been mitigated and the outcomes documented. Root cause is assessed and addressed as part of the incident response process.

CSIRT and PSIRT also document Legal engagement and external notification processes in the event an incident is validated as a breach or a violation of contractual commitments. GitHub informs affected users and organizations as required. The messaging can be sent via various forms such as email, a changelog post, a blog post, or release notes. Investigations specific to an affected customer are managed directly with that customer, in accordance with contractual terms.

Change Management

Change Management Overview

GitHub is a collaborative, organic, and adaptable organization. GitHub Enterprise Cloud service's code and system change management processes leverage GitHub Enterprise Cloud service's native source code and changelog technology available to users and organizations.

Change Development

GitHub application and configuration changes are developed and maintained within GitHub's private enterprise on the GitHub Enterprise Cloud service product. Individual engineers are grouped by teams based on their areas of code ownership and expertise. Access to branch, deploy, approve, and merge changes to critical GitHub repositories is restricted to members of the Platform Health and Production Engineering teams. Engineers responsible for the critical GitHub Enterprise Cloud service repositories of the codebase receive timely notifications of changes pushed and monitor these changes through automated issues posted in the repositories and in Slack.

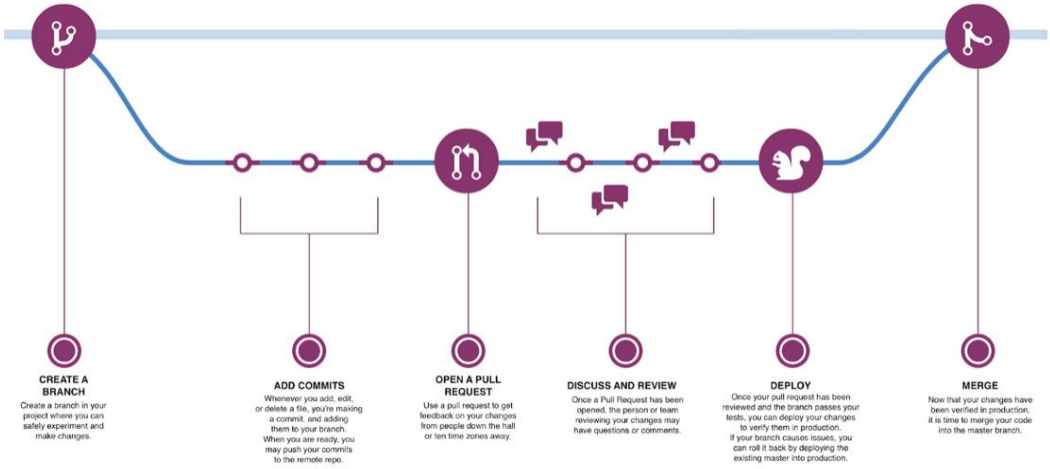

Figure 3: Change Development Diagram

When developing code for GitHub Enterprise Cloud service, engineers create a branch from the main branch, which is the "gold copy" of GitHub, to begin development of changes. Depending on the change, developers test locally using automated CI/CD testing described below. Once pre-production testing has validated the change is functioning as expected and all required reviews and approvals are complete, the branch is merged into the main branch and is rolled out in staged releases via canary deployments to GitHub's Kubernetes clusters in production for customer use.

An automated static code analysis tool runs every time new code is committed to detect insecure coding practices based on third-party-provided lexical, syntactic, and semantic rulesets, in addition to GitHub-

developed rulesets and test scenarios. An automated alerting issue is posted on the containing issue and in a tracking repository maintained by the Product Security Engineering team to alert them of any potentially insecure coding risks to product development. The developer assesses the potential vulnerabilities, and a peer reviewer follows up on the items during change review and approval.

Peer Review

GitHub relies heavily on collaboration and peer reviews to help ensure the integrity of its products. Requirements for performing an effective peer review are documented on The Hub and in repository-specific pull request templates. Once the code is ready for discussion and review, developers open a pull request for commits added to their branch following the template. Pull requests are an integral feature of GitHub used to coordinate the discussion, approval, testing, and deployment of changes.

Protected branches (branches where merging to the main branch is gated by required review and approval activities) are also enabled on GitHub Enterprise Cloud service code repositories. Protected branches require pull requests to be reviewed and approved by an individual who is different than the proposer of the change prior to merging the code to the main branch.

The Security team conducts high-level design reviews, including security reviews, and rapid risk assessments for large projects and initiatives to assess security risk. Depending on the scope and type of project or initiative, the appropriate security teams are engaged to review the project and provide recommendations and guidance to Engineering and Product Management. Recommendations are tracked in GitHub issues, and teams must approve plans prior to release.

Automated Testing

CI is the process by which code changes are automatically tested and validated. When a change is committed to a branch with an associated pull request, the CI system automatically executes the configured CI status checks for that repository. This process includes the following steps:

- CI is notified of changes to the repository and builds the code for testing.
- CI runs the test suites selected for that repository, including end-to-end integration testing, functional/unit testing, and security testing.
- CI successes and failures are referenced in the pull request.

Failures in CI will block merging and deploying of those changes. CI testing occurs every time a change is committed and when a change is deployed to production. GitHub CI testing uses a suite of tools to automate build and test on isolated hosts. Test results are automatically exported and alerted on the respective issues, allowing teams to monitor and act on issues prior to deployment.

GitHub engineers collaborate to support the creation of CI test scripts and the implementation of blocking acceptance tests. These acceptance tests typically function at the browser level and focus on the overall functionality of the module or application to help ensure customer commitments and system requirements are met. GitHub prevents any change from deployment to production if required acceptance tests have not passed.

Deployment

To support deployment to the production environment, changes are deployed using GitHub's standard and proprietary deployment tools and workflows. To deploy any application or configuration changes, developers run a Slack ChatOps deployment command on the respective team channels associated with the production application. Running the Slack ChatOps creates a traceable record of the action and helps to ensure accountability and transparency within the team and other stakeholders subscribed to the respective Slack channel. Automated deployment processes validate that required checks are completed, including passing of mandatory acceptance checks and ensuring the most recent main branch build is used,

before deploying the change to production. Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification.

Monitoring, Metrics, and Change Rollback

To help ensure continued functionality, availability, and security of the GitHub Enterprise Cloud service system, comprehensive logging and monitoring tools are built into the application, system, and network to monitor real-time telemetry, such as network traffic, successful logins, exceptions and error messages from the application and systems, and pseudonymized usage statistics to detect anomalies through fine-tuned fault tolerances and historical trend analyses.

Engineering teams monitor each system component post-deployment to help ensure changes implemented did not detrimentally impact stability. A dedicated Site Reliability Engineering team monitors and responds to incidents 24/7 through real-time, automated monitoring and incident escalation workflow. In the event of any detrimental impact to production resulting from a change, changes are rolled back to the last known stable version of the main branch. As a matter of practice, GitHub uses branch deployments to production, so the roll-back procedure is simple and can be performed by anyone with authorization to deploy.

Once a deployment to production is shown to be faulty or unstable, based on the continuous monitoring described previously, manual testing, or some other measure, the assigned engineer runs a single ChatOps command, and the previous version of the main branch is deployed back to production. Depending on the severity of the findings, the engineer can make immediate changes and redeploy, or unlock deploys and begin testing again in limited environments.

To keep GitHub personnel in the loop on new feature development or other engineering efforts, the Product Management teams regularly host internal demos for the organization as a part of regular company and IT all-Company get-togethers. These presentations are a way for IT teams to discuss and share projects implemented and goals achieved throughout the quarter.

New customer-impacting product and feature releases, defined as “notable changes,” go through a standardized release process to help ensure the appropriate level of communication given the type of change. A notable change is anything added, changed, deprecated, removed, or fixed, or any security fixes or Common Vulnerabilities and Exposures (CVEs) that customers should understand. These changes could affect the productivity or workflow of the user, or in the case of security fixes, require notification and awareness for customers. For example, notable changes include API deprecations, user interface (UI) improvements, or additions to payment options. These changes are communicated at several discrete stages of development and prior to launch. Methods of communication include blog posts and may also include social media posts and product training videos.

Emergency Deploys

When an emergency or system failure means a deploy cannot follow the normal procedures, there is an established process to allow engineers to circumvent the usual tooling and force a deployment into production. The emergency deploy process, and knowing when it is justified, is a key component within GitHub culture.

The ability to perform an emergency deploy is configured to be restricted to a limited group of engineers (including members in the Incident Commander rotation, along with a subset group of select engineers who may need to be involved in incidents) through assignment to the “gh-force-deploy” entitlement. This group is allowed to force deploy changes; in addition, they can temporarily grant the permission to other users on an as-needed basis for 30 minutes. Emergency deploys are requested in the same pre-defined deploy Slack channels used for regular deploys, and the engineer requesting the deploy specifies a deploy reason to give context to the situation to observers and others who are monitoring these deploys.

After an engineer force deploys, the deploy queue is blocked for that repository until the engineer reverts the changes or merges the changes into the main branch. Both paths require CI and peer review and approval to complete, to ensure the same level of visibility and testing as any other changes to the code base.

When an emergency deploy is executed, an alert is posted to the #incident-command Slack channel for visibility to the on-call team that is on standby via Pager Duty rotations. A postmortem remediation issue is auto-created in the github/forced-deploys repository. The issue is assigned to the engineer who deployed along with a postmortem checklist the assignee completes within 24 hours, giving justification for the deploy, which is then reviewed by their manager and a Site Reliability Engineering team member. In the event of inactivity, automatic escalations occur to bring attention to the stale issue.

Postmortem remediation issues are reviewed weekly in the Production Engineering Availability team meeting. The deploying engineer is invited to the meeting to review the issue and the circumstances, and to verify they have completed the expected post-deploy steps to help ensure no further issues require resolution.

Complementary User Entity Controls (CUECs)

GitHub's controls related to GitHub Enterprise Cloud service cover only a portion of overall internal control for each user entity of GitHub Enterprise Cloud service. It is not feasible for the service commitments, system requirements, and applicable criteria related to the system to be achieved solely by GitHub. Therefore, each user entity's internal control should be evaluated in conjunction with GitHub's controls and the related tests and results described in Section 4 of this report, taking into account the related CUECs identified for the specific criterion. In order for user entities to rely on the controls reported herein, each user entity must evaluate its own internal control to determine whether the identified CUECs have been implemented and are operating effectively.

The CUECs presented should not be regarded as a comprehensive list of all controls that should be employed by user entities. Management of user entities is responsible for the following:

Control Objective	Complementary User Entity Controls
CO3	• User entities deploy physical security and environmental controls for all devices and access points residing at their operational facilities, including remote employees or at-home agents for which the user entity allows connectivity.
CO4	• Customer administrators are responsible for setting their authentication policy for GitHub Enterprise Cloud service user accounts. GitHub recommends the use of SAML/SSO with the customer's identity provider. • Enabling 2FA and ensuring members and collaborators require 2FA; this includes the implementation and management of personal access tokens.
CO5	• It is the responsibility of the user entity to have policies and procedures to: – Inform their employees and users that their information or data is being used and stored by GitHub. – Determine how to file inquiries, complaints, and disputes to be passed on to GitHub.
CO6	• Administering and configuring repositories, including permissions, enabling required reviews for pull requests, and enabling required status checks before merging. • Securing configuration of GitHub Actions. See https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions for detailed guidance.

Subservice Organizations and Complementary Subservice Organization Controls (CSOCs)

GitHub uses CoreSite, Sabey, Equinix, and QTS as subservice organizations for data center colocation services and Azure and AWS for infrastructure hosting. GitHub's controls related to GitHub Enterprise Cloud service cover only a portion of the overall internal control for each user entity of GitHub Enterprise Cloud service. The description does not extend to the colocation services for IT infrastructure provided by the subservice organizations. Section 4 of this report and the description of the system only cover the Trust Services Criteria and related controls of GitHub and exclude the related controls of Azure, AWS, CoreSite, Sabey, Equinix, and QTS.

Although the subservice organizations have been carved out for the purposes of this report, certain service commitments, system requirements, and applicable criteria are intended to be met by controls at the subservice organizations. CSOCs are expected to be in place at Azure, AWS, CoreSite, Sabey, Equinix, and QTS related to physical security and environmental protection. The subservice organizations' physical security controls should mitigate the risk of unauthorized access to the hosting facilities. The subservice organizations' environmental protection controls should mitigate the risk of fires, power loss, climate, and temperature variabilities.

Company management receives and reviews the Azure, AWS, CoreSite, Sabey, Equinix, and QTS SOC 2 reports, ISO 27001 and 27701 Certifications, and Payment Card Industry Attestation of Compliance (PCI AOC) as they are issued and at least annually. In addition, through its operational activities, Company management monitors the services performed by Azure, AWS, CoreSite, Sabey, Equinix, and QTS to determine whether operations and controls expected to be implemented are functioning effectively. Management also communicates with the subservice organizations to monitor compliance with the service agreements, stay informed of changes planned at the hosting facilities, and relay any issues or concerns to Azure, AWS, CoreSite, Sabey, Equinix, and QTS management.

It is not feasible for the service commitments, system requirements, and applicable criteria related to GitHub Enterprise Cloud service to be achieved solely by GitHub. Therefore, each user entity's internal control must be evaluated in conjunction with GitHub's controls and related tests and results described in Section 4 of this report, taking into account the related CSOCs expected to be implemented at Azure, AWS, CoreSite, Sabey, Equinix, and QTS as described below.

Control Objective	Complementary Subservice Organization Controls
Network Security	• Azure and AWS encrypt customer data at rest. • Azure and AWS correct functioning of tenant isolation of their cloud services.
Physical Security	• Azure, AWS, CoreSite, Sabey, Equinix, and QTS restrict data center access to authorized personnel. • Azure, AWS, CoreSite, Sabey, Equinix, and QTS monitor data centers 24/7 by closed circuit cameras and security personnel.
Logical and Physical Access	• Azure and AWS securely decommission and physically destroy production assets in their control.

Control Objective	Complementary Subservice Organization Controls
Environmental Protections	• Azure, AWS, CoreSite, Sabey, Equinix, and QTS install fire suppression and detection and environmental monitoring systems at the data centers. • Azure, AWS, CoreSite, Sabey, Equinix, and QTS protect data centers against a disruption in power supply to the processing environment by an uninterruptible power supply (UPS). • Azure, AWS, CoreSite, Sabey, Equinix, and QTS oversee the regular maintenance of environmental protections at the data centers.

Significant Changes to the System

GitHub began using AWS for infrastructure hosting services in October 2024.

There were no other changes that are likely to affect report users' understanding of how GitHub Enterprise Cloud service was used to provide the service from October 1, 2024 to March 31, 2025.

Report Use

The description does not omit or distort information relevant to GitHub Enterprise Cloud service while acknowledging that the description is prepared to meet the common needs of a broad range of users and may not, therefore, include every aspect of the system that each individual user may consider important to their own particular needs.

Section 4

Description of GitHub, Inc.'s Control Objectives and Related Controls, and Independent Service Auditor's Description of Tests of Controls and Results

Control Environment Elements

The control environment represents the collective effect of various elements in establishing, enhancing or mitigating the effectiveness of specific controls. The control environment elements as described in the description of the system include, but are not limited to, the Handbook for Hubbers, Policies and Procedures and Human Resources.

Our tests of the control environment included the following procedures, to the extent we considered necessary; (a) an inspection of GitHub's organizational structure including segregation of functional responsibilities and policies and procedures; (b) inquiries with management, operations, administrative and other personnel who are responsible for developing, ensuring adherence to and applying controls; (c) observations of personnel in the performance of their assigned duties; and (d) inspection of documents and records pertaining to controls.

Description of Tests Performed by Coalfire Controls, LLC

Our tests of operating effectiveness of controls included such tests as were considered necessary in the circumstances to evaluate whether those controls, and the extent of compliance with them, were sufficient to provide reasonable, but not absolute, assurance that the control objectives were achieved throughout the period October 1, 2024 to March 31, 2025. In selecting particular tests of the operating effectiveness of the controls, we considered (i) the nature of the controls being tested; (ii) the types of available evidential matter; (iii) the nature of the control objective to be achieved; (iv) the assessed level of control risk; and (v) the expected efficiency and effectiveness of the test. Such tests were used to evaluate fairness of the presentation of the description of GitHub's information technology general control system for the GitHub Enterprise Cloud service and to evaluate the operating effectiveness of specified controls.

Additionally, observation and inspection procedures were performed as it relates to system generated reports, queries, and listings within management's description of the system to assess the completeness and accuracy (reliability) of the information utilized in the performance of our testing of the control activities.

The controls in the second column in the following tables and the control objectives detailed at the top of each table are part of management's description of the service organization's system that appears in Section 3 of this report. The description of tests of controls and results in the third and fourth columns, respectively, are provided by the service auditor.

Control Objective 1: Organization and Management ***Controls provide reasonable assurance that human resources policies and procedures address accountability, integrity, ethical values and qualifications of personnel, and the environment in which they function.***			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
1.1	Individuals with access to the in-scope environment acknowledge and sign off on the Handbook for Hubbers as part of the onboarding process. The Handbook includes the GitHub standards of conduct, Security Policy Awareness and Responsibilities, and other information security topics.	Inspected the Handbook for Hubbers to determine that it described the GitHub standards of conduct, Security Policy Awareness and Responsibilities, and other information security topics.	No exceptions noted.
		Inspected acknowledgements for a sample of new employees with access to the in-scope environment to determine that new employees acknowledged that they had read and agreed to the Handbook for Hubbers as part of the onboarding process.	No exceptions noted.
1.2	Employees sign a confidentiality agreement upon hire. This agreement prohibits the disclosure of information and other data to which the employee has been granted access during employment and after termination.	Inspected the confidentiality agreement template to determine that it prohibited the disclosure of information and other data to which the employee had been granted access during employment and after termination.	No exceptions noted.
		Inspected signed confidentiality agreements for a sample of new employees to determine that new employees were provided and signed the agreement upon hire.	No exceptions noted.
1.3	GitHub management evaluates and provides feedback to direct reports annually.	Inspected performance appraisal documentation for a sample of employees to determine that management evaluated and provided feedback to direct reports during the period.	No exceptions noted.

Control Objective 1: Organization and Management ***Controls provide reasonable assurance that human resources policies and procedures address accountability, integrity, ethical values and qualifications of personnel, and the environment in which they function.***			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
1.4	The Company has documented disciplinary actions within the Handbook for Hubbers for employees and contractors who violate the code of conduct.	Inspected the Handbook for Hubbers to determine that the Company had documented disciplinary actions within the Handbook for Hubbers for employees and contractors who violated the code of conduct.	No exceptions noted.
1.5	New employees and contractors undergo background screening as permissible by local laws and regulations.	Inspected background check completion evidence for a sample of new employees and contractors to determine that new employees and contractors underwent background screening as permissible by local laws and regulations.	No exceptions noted.
1.6	The Senior Leadership team sets Company-wide objectives and reviews key results semiannually. Operational objectives and key results defined across the in-scope domains tie back to meeting those company-wide organizational objectives, including security and risk objectives. The Senior Leadership team semiannually reports and reviews progress towards meeting those objectives.	Inspected the semiannual objectives set by the Senior Leadership team to determine that the Senior Leadership team set Company-wide objectives, reviewed the key results, and reported and reviewed progress towards meeting those objectives during the period and that operational objectives and key results defined across the in-scope domains tied back to meeting those Company-wide objectives.	No exceptions noted.
1.7	People Operations maintains a current organizational chart that outlines Hubbers' roles and is made available to employees and contractors.	Inspected the organizational chart to determine that it outlined Hubbers' roles and was made available to employees and contractors.	No exceptions noted.

Control Objective 1: Organization and Management ***Controls provide reasonable assurance that human resources policies and procedures address accountability, integrity, ethical values and qualifications of personnel, and the environment in which they function.***			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
1.8	The GitHub Information Security Management System (ISMS) defines information security practices, including information system management, access controls, software development, and the definition of roles and responsibilities. This policy is published to internal personnel via the GitHub intranet. The Chief Information Security Officer (CISO) reviews and approves the policy annually.	Inspected the GitHub ISMS to determine that the CISO reviewed and approved the policy during the period and that it defined information security practices including information system management, access controls, software development, and the definition of roles and responsibilities.	No exceptions noted.
		Inspected the Company intranet to determine that the GitHub ISMS was published to internal personnel via the GitHub intranet.	No exceptions noted.
1.9	Hubber responsibilities are communicated through documented job descriptions, which are stored internally.	Inspected job descriptions for a sample of employees supporting the service to determine that Hubber responsibilities were communicated through documented job descriptions and were stored internally.	No exceptions noted.
1.10	Hubbers and contractors complete security and privacy awareness training within 45 days of their start date and Hubbers complete security and privacy awareness training on an annual basis.	Inspected training completion evidence for a sample of new Hubbers and contractors to determine that new Hubbers and contractors completed security and privacy awareness training within 45 days of their start date.	Exceptions noted. 3 out of a sample of 19 new Hubbers did not complete security and privacy awareness training within the stipulated deadline. However, all new contractors out of a sample of 6 new contractors did complete security and privacy awareness training within the stipulated deadline.

Control Objective 1: Organization and Management *Controls provide reasonable assurance that human resources policies and procedures address accountability, integrity, ethical values and qualifications of personnel, and the environment in which they function.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
		Inspected training completion evidence for a sample of active Hubbers to determine that Hubbers completed security and privacy awareness training according to the normal cadence in July and August 2024.	No exceptions noted. The security and privacy awareness training was completed according to the normal cadence in July and August 2024.
1.11	The Handbook for Hubbers outlines the process for anonymously reporting potential security issues or fraud concerns.	Inspected the Handbook for Hubbers to determine that it outlined the process for anonymously reporting potential security issues or fraud concerns.	No exceptions noted.

Control Objective 2: Risk Management

Controls provide reasonable assurance that the Company (i) identifies potential risks that would affect the entity's ability to achieve its objectives, (ii) analyzes those risks, (iii) develops responses to those risks including the design and implementation of controls and other risk mitigating actions, and (iv) conducts ongoing monitoring of risks and the risk management process.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
2.1	The Company specifies its objectives in its annual risk assessment to enable the identification and assessment of risk related to the objectives.	Inspected documentation from the risk assessment performed during the period to determine that the Company specified its objectives in its annual risk assessment to enable the identification and assessment of risk related to the objectives.	No exceptions noted.
2.2	GitHub has established a formal Security Risk Management Policy to identify, assess, and manage security and security compliance-related risks. GitHub Security leadership reviews and approves the Security Risk Management Policy annually.	Inspected the Security Risk Management Policy to determine that a program had been established to identify, assess, and manage security and security compliance-related risks.	No exceptions noted.
		Inspected the Security Risk Management Policy to determine that it was reviewed and approved by the GitHub Security leadership according to the normal cadence and completed in June 2024.	No exceptions noted. The Security Risk Management Policy was reviewed according to the normal cadence and completed in June 2024.
2.3	Annually, GitHub performs a risk assessment that identifies, tracks, and monitors changes to in-scope systems security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution.	Inspected risk assessment documentation performed during the period to determine that it identified, tracked, and monitored changes to in-scope systems security, related compliance risk factors, and fraud, and that risks identified were documented in a repository where the risks and mitigation actions were tracked to resolution.	No exceptions noted.

Control Objective 2: Risk Management

Controls provide reasonable assurance that the Company (i) identifies potential risks that would affect the entity's ability to achieve its objectives, (ii) analyzes those risks, (iii) develops responses to those risks including the design and implementation of controls and other risk mitigating actions, and (iv) conducts ongoing monitoring of risks and the risk management process.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
2.4	Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repository. Annually, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports.	Inspected new vendor security compliance documentation and the vendor security repository for all new vendors to determine that GitHub reviewed vendor security compliance documentation and documented the results in the vendor security repository.	No exceptions noted.
		Inspected subservice organization risk assessment documentation for all subservice organizations to determine that GitHub assessed the controls implemented at subservice organizations that received or stored customer data by reviewing the relevant attestation reports according to the normal cadence and completed in May and June 2024.	No exceptions noted. The subservice organization risk assessment was documented according to the normal cadence and completed in May and June 2024.
2.5	As part of its annual risk assessment, management selects and develops manual and IT general control activities that contribute to the mitigation of identified risks.	Inspected documentation from the risk assessment performed during the period to determine that, as part of its annual risk assessment, management selected and developed manual and IT general control activities that contributed to the mitigation of identified risks.	No exceptions noted.

Control Objective 3: Network Security ***Controls provide reasonable assurance that GitHub actively monitors the network, maintains network security standards, and prevents unauthorized access to customer data.***			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
3.1	GitHub's Security and Infrastructure teams subscribe to industry security bulletins and email alerts and use them to monitor the impact of emerging technologies and security on the production systems.	Inspected example security bulletins and email alerts to determine that GitHub's Security and Infrastructure teams subscribed to industry security bulletins and email alerts and used them to monitor the impact of emerging technologies and security on the production systems.	No exceptions noted.
3.2	GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes.	Inspected the GitHub intranet to determine that GitHub maintained and communicated key security standards and procedures on the GitHub intranet that addressed the security of systems, facilities, data, personnel, and processes.	No exceptions noted.
3.3	GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures.	Inspected hardening standards to determine that they were documented based on the Company's security hardening practices and that the Security Operations team reviewed and approved changes to the standard procedures during the period.	No exceptions noted.
3.4	Customer Git repositories are encrypted at rest.	Inspected the configurations for an example customer Git repository to determine that customer Git repositories were encrypted at rest.	No exceptions noted.
3.5	Virtual Private Network (VPN) and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.	Inspected the VPN and bastion host configurations to determine that they required user connections to reauthenticate after 24 hours of inactivity.	No exceptions noted.

Control Objective 3: Network Security

Controls provide reasonable assurance that GitHub actively monitors the network, maintains network security standards, and prevents unauthorized access to customer data.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
3.6	GitHub uses SSH to authenticate to back-end production resources through a bastion host.	Inspected system configurations and observed login attempts to back-end production resources to determine that GitHub used SSH to authenticate to back-end production resources through a bastion host.	No exceptions noted.
3.7	Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary.	Inspected network firewall configurations to determine that network firewalls were used and that they restricted external points of connectivity and prevented unauthorized traffic from beyond the system boundary.	No exceptions noted.
3.8	Firewall rulesets are reviewed at least annually. Change tickets are created to track any firewall modifications as a result of the review.	Inspected the firewall review documentation to determine that firewall rulesets were reviewed during the period.	No exceptions noted.
		Inspected change tickets for a sample of changes that resulted from the firewall ruleset review to determine that change tickets were created to track firewall modifications resulting from the review.	No exceptions noted.
3.9	Production traffic is encrypted when transmitted over the public internet.	Inspected transmission protocol configurations to determine that secure data transmission protocols were used to encrypt production data when transmitted over the public internet.	No exceptions noted.

Control Objective 3: Network Security *Controls provide reasonable assurance that GitHub actively monitors the network, maintains network security standards, and prevents unauthorized access to customer data.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
3.10	An intrusion detection system (IDS) is used to provide continuous monitoring of the Company's network and early detection of potential security breaches. Alerts are configured to notify administrators to investigate and take appropriate action based on the severity of the alert.	Inspected IDS configurations to determine that an IDS was used to provide continuous monitoring of the Company's network and early detection of potential security breaches.	No exceptions noted.
		Inspected IDS alert configurations and example alerts to determine that alerts were configured to notify administrators to investigate and take appropriate action based on the severity of the alert.	No exceptions noted.
3.11	Threat detection software is installed on Company production servers and endpoints to monitor for malicious software or unauthorized activity.	Inspected threat detection software configurations to determine that threat detection software was installed on Company production servers and endpoints to monitor for malicious software or unauthorized activity.	No exceptions noted.
3.12	GitHub security personnel are informed of security events, whether detected by systems or reported by humans, and alerts are surfaced to responders based on severity as defined in formally documented procedures.	Inspected tickets for a sample of security events to determine that GitHub security personnel were informed of security events, whether detected by systems or reported by humans, and alerts were surfaced to responders based on severity as defined in formally documented procedures.	No exceptions noted.

Control Objective 3: Network Security

Controls provide reasonable assurance that GitHub actively monitors the network, maintains network security standards, and prevents unauthorized access to customer data.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
3.13	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents. The guidance also allows the Company to address breach or privacy considerations, which vary depending on the severity of the incident.	Inspected the security incident response and escalation policies and procedures to determine that they were documented and provided guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and addressed breach or privacy considerations depending on the severity of the incident.	No exceptions noted.
3.14	All incidents related to security and involving a data breach are logged, tracked, evaluated, and communicated to affected parties by management until the Company has recovered from the incidents. The control did not operate during the period because the circumstances that warrant the operation of the control did not occur during the period. No security incidents occurred during the period.	Inquired of management and inspected security event documentation to determine that the circumstances that warrant the operation of the control did not occur during the period. As a result, no testing could be performed to determine whether all incidents related to security and involving a data breach were logged, tracked, evaluated, and communicated to affected parties by management until the Company had recovered from the incidents.	Not tested. No security incidents were identified during the period.
3.15	GitHub has incident response procedures that are triggered once an incident is identified or reported. GitHub has an escalation procedure to address breach or privacy considerations, which varies depending on the severity of the incident. Roles are assigned, and a multi-stage process is followed until the incident has been	Inspected the incident response procedures to determine that GitHub had incident response procedures in place that were triggered when an incident was identified or reported and had an escalation procedure in place to address breach or privacy considerations depending on the severity of the incident.	No exceptions noted.

Control Objective 3: Network Security *Controls provide reasonable assurance that GitHub actively monitors the network, maintains network security standards, and prevents unauthorized access to customer data.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
	remedied and documented. Root cause is assessed and addressed as part of the incident response process. Incident post-mortems are documented and include lessons learned. A portion of the control did not operate during the period because the circumstances that warrant the operation of the control did not occur during the period. No security incidents involving a data breach were identified during the period.	Inquired of management and inspected security event documentation to determine that the circumstances that warrant the operation of this portion of the control did not occur during the period. As a result, no testing could be performed to determine whether security incident documentation followed the incident response procedures and escalation procedure, roles were assigned to the incident, a multi-stage process was followed until the incident had been remedied and documented, the root cause was assessed and addressed, and the incident post-mortems were documented and included lessons learned.	Not tested. No security incidents were identified during the period.

Control Objective 4: Logical Access ***Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.***			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
4.1	Formal procedures are documented that outline the process the Company's staff follows to perform the following system access control functions: - Authorizing and granting new user access - Maintaining existing user access - Managing entitlements - Revoking existing user access - Restricting access based on entitlement membership and access request approval	Inspected system access control procedures to determine that formal procedures were documented that outlined the process the Company's staff followed to perform the following system access control functions: - Authorizing and granting new user access - Maintaining existing user access - Managing entitlements - Revoking existing user access - Restricting access based on entitlement membership and access request approval	No exceptions noted.
4.2	Authentication to the following in-scope production system components requires unique usernames and passwords or authorized Secure Shell (SSH) keys: - Network - Operating system (OS) - Application(s) - Data stores - Azure and Amazon Web Services (AWS) consoles - Firewalls - Log data - Backup data	Inspected system configurations and observed login attempts to determine that authentication to the following in-scope production system components required unique usernames and passwords or authorized SSH keys: - Network - OS - Application(s) - Data stores - Azure and AWS consoles - Firewalls - Log data - Backup data	No exceptions noted.

Control Objective 4: Logical Access *Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
4.3	A unique account and password are required to authenticate customers to their organization for in-scope systems.	Inspected system and Lightweight Directory Access Protocol (LDAP) configurations and observed login attempts into the in-scope environment and s to determine that a unique account and password were required to authenticate customers to their organization.	No exceptions noted.
4.4	Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token.	Inspected system configurations and observed remote login attempts to internal administration tools to determine that remote access to internal administration tools was restricted through unique account IDs, passwords, and a one-time token.	No exceptions noted.
4.5	Remote access to in-scope production systems is restricted through VPN, bastion hosts, or Security Assertion Markup Language (SAML) enforcing Hypertext Transfer Protocol Secure (HTTPS) reverse proxy, which require multi-factor authentication (MFA).	Inspected system configurations and observed login attempts to in-scope production systems to determine that remote access to in-scope production systems was through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which required MFA.	No exceptions noted.
4.6	Passwords for in-scope system components are configured according to the Company's policy, which requires the following (unless there is a system limitation): - Eight characters long, if it includes a number and a lowercase letter or - 15 characters long with any combination of characters	Inspected the password policy and password configurations for in-scope system components to determine that passwords were configured according to Company policy, which required the following (unless there was a system limitation): - Eight characters long, if it includes a number and a lowercase letter or - 15 characters long with any combination of characters	No exceptions noted.

Control Objective 4: Logical Access *Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
4.7	Test environments are logically separated from production environments.	Inspected the test and production environments to determine that they were logically separated from each other.	No exceptions noted.
4.8	GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role and require manager approval prior to being provisioned. GitHub's internal systems are configured to automatically provision non-role-based entitlements only after the user's manager reviews and approves the access request.	Inspected management approval evidence for a sample of access provisioning requests to determine that access was provisioned based on the user's role and management approval was required prior to provisioning logical and physical access.	No exceptions noted.
		Inspected access provisioning configurations within internal systems to determine that they were configured to automatically provision logical and physical access based on the user's job role and non-role-based entitlements only after the request has been reviewed and approved.	No exceptions noted.
4.9	The Security Operations team revokes logical production access, and Production Engineering revokes physical production access for terminated personnel (employee and contractors) within 24 hours of termination.	Inspected documentation for a sample of terminated employees to determine that a termination checklist was completed and logical and physical access was revoked by the Security Operations and Production Engineering team within 24 hours of termination as part of the termination process.	Exceptions noted. 2 out of a sample of 25 terminated employees and contractors did not have logical and physical access revoked within 24 hours of termination.

Control Objective 4: Logical Access *Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
		Inspected a listing of terminated employees and contractors and compared the listing to the active in-scope system access listings to determine that terminated employees and contractors did not retain logical access to the in-scope systems after their separation.	No exceptions noted.

Control Objective 4: Logical Access *Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
4.10	Semiannually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm that users with this access are restricted based on the principle of least privilege. Tickets are created to remove or modify access as necessary in a timely manner.	Inspected access review documentation performed during the period to determine that semiannual access reviews were conducted by the Security Operations team to confirm that users with elevated access permissions were restricted based on the principle of least privilege.	No exceptions noted.
		Inspected change tickets for a sample of changes that resulted from the access review to determine that change tickets were created to remove or modify access as necessary in a timely manner.	No exceptions noted.
4.11	The Infrastructure team reviews physical access to production data centers annually; any unauthorized accounts are removed. A portion of the control did not operate during the period because the circumstances that warrant the operation of the control did not occur during the period. No users required access to be revoked or modified during the period.	Inspected the production data centers physical access review documentation from the period to determine that an annual physical access review was conducted by the Infrastructure team to help ensure that access was restricted appropriately.	No exceptions noted.
		Inquired of management and inspected production data centers physical access review and account removal documentation to determine that the circumstances that warrant the operation of the control did not occur during the period. As a result, no testing could be performed to determine whether unauthorized accounts were removed as a result of the Infrastructure team's annual review of physical access to production data centers.	Not tested. No users were identified that required access to be revoked or modified during the period.

Control Objective 4: Logical Access *Controls provide reasonable assurance that GitHub restricts logical access to the system, provides and removes that access, and prevents unauthorized access.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
4.12	Privileged access to the following in-scope production system components is restricted to authorized users with a business need: - Network - OS - Application(s) - Data stores - Azure and AWS consoles - Firewalls - Log data - Backup data - Encryption keys	Inspected the access listings, inquired of management, and compared each user's level of access to their job role to determine that privileged access to the following in-scope production system components was restricted to authorized users with a business need: - Network - OS - Application(s) - Data stores - Azure and AWS consoles - Firewalls - Log data - Backup data - Encryption keys	No exceptions noted.

Control Objective 5: System Operations

Controls provide reasonable assurance that GitHub manages the execution of system procedures and detects and mitigates deviations and issues.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
5.1	The Company maintains a Bug Bounty Program that offers rewards for discovered vulnerabilities and is designed to ensure the security and integrity of the platform through treating these findings.	Inspected a sample of Bug Bounty tickets to determine that the Company maintained a Bug Bounty Program that offered rewards for discovered vulnerabilities and was designed to ensure the security and integrity of the platform and that findings were ticketed and treated.	No exceptions noted.
5.2	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.	Inspected internal and external network vulnerability scans for a sample of months to determine that the Security Operations team scanned internal and external systems monthly, reviewed identified vulnerabilities, and shared confirmed threats with responsible stakeholders.	No exceptions noted.
5.3	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.	Inspected remediation plans for vulnerabilities identified during the sampled monthly internal and external vulnerability scans to determine system owners remediated vulnerabilities in accordance with the established SLAs based on the severity of the vulnerability, and any vulnerabilities that were unable to be remediated within the required SLA were handled via the documented exception process.	Exceptions noted. 2 out of a sample of 44 vulnerabilities identified during the sampled monthly internal and external vulnerability scans that were unable to be remediated within the required SLA were not handled via the documented exception process.

Control Objective 5: System Operations *Controls provide reasonable assurance that GitHub manages the execution of system procedures and detects and mitigates deviations and issues.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
5.4	A configuration management tool (CMT) maintains the state of systems to an approved baseline and reverts unapproved changes, when they are detected, back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review.	Inspected CMT configurations to determine that a CMT was in place to maintain the state of systems to an approved baseline and revert unapproved changes, when they are detected, back to the required configuration.	No exceptions noted.
		Inspected a sample of changes to the approved configuration baseline to determine that they were completed through pull requests and required a peer review.	No exceptions noted.
5.5	Logging and monitoring tools are used to collect telemetry and event logs from network devices, applications, and systems. Security alerts are generated, prioritized, and communicated to the designated responders for follow up. GitHub Security teams use these alerts to initiate triage and, if necessary, remediate security issues based on their assigned priority.	Inspected the log management tool configurations to determine that logging and monitoring tools were used to collect telemetry and event logs from network devices, applications, and systems.	No exceptions noted.
		Inspected the log management tool alert configurations and tracking documentation to determine that security alerts were generated, prioritized, and communicated to the designated responders for follow up, and GitHub Security teams used alerts to initiate triage and, if necessary, remediated security issues based on their assigned priority.	No exceptions noted.

Control Objective 5: System Operations *Controls provide reasonable assurance that GitHub manages the execution of system procedures and detects and mitigates deviations and issues.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
5.6	An infrastructure and application monitoring tool is utilized to monitor system or infrastructure availability and performance and generates alerts when specific, predefined thresholds are met.	Inspected the infrastructure and application monitoring tool configurations and example alerts to determine that an infrastructure and application monitoring tool was utilized to monitor system or infrastructure availability and performance and generated alerts when specific, predefined thresholds were met.	No exceptions noted.
5.7	A third party performs an annual application penetration assessment on components of the in-scope system. Security Operations triages and tracks identified issues through to resolution.	Inspected the penetration test report to determine that a third party performed an application penetration assessment, on components of the in-scope system, during the period.	No exceptions noted.
		Inspected remediation plans for vulnerabilities identified during the penetration test to determine that the Security Operations team triaged and tracked identified issues through to resolution.	No exceptions noted.
5.8	The GitHub external website communicates to internal and external users the description of the in-scope systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints.	Inspected the GitHub external website to determine that it communicated to internal and external users the description of the in-scope systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints.	No exceptions noted.

Control Objective 6: Change Management *Controls provide reasonable assurance that GitHub identifies the need for changes to the system, makes the changes following a controlled change management process, and prevents unauthorized changes from being made.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
6.1	Formal policies and procedures that outline the technical and organizational safeguards for change management of system components are documented and include the following components: - Change management roles and responsibilities - Criteria for risk assessment, categorization, and prioritization of changes - Approvals for implementation of changes - Requirements for the performance and documentation of tests, including rollback plans - Requirements for segregation of duties during development, testing, and release of changes - Requirements for the implementation and documentation of emergency changes	Inspected the change management procedures to determine that formal policies and procedures that outlined the technical and organizational safeguards for change management of system components were documented and included the following components: - Change management roles and responsibilities - Criteria for risk assessment, categorization, and prioritization of changes - Approvals for implementation of changes - Requirements for the performance and documentation of tests, including rollback plans - Requirements for segregation of duties during development, testing, and release of changes - Requirements for the implementation and documentation of emergency changes	No exceptions noted.
6.2	A formal security and software development life cycle (SDLC) methodology is in place that governs the project planning, design, acquisition, testing, implementation, maintenance, and decommissioning of information systems and related technologies.	Inspected security and SDLC documentation to determine that a formal security and SDLC methodology was in place that governed the project planning, design, acquisition, testing, implementation, maintenance, and decommissioning of information systems and related technologies.	No exceptions noted.
6.3	An independent reviewer evaluates and approves software and infrastructure change requests via a GitHub pull request.	Inspected a sample of pull requests for software and infrastructure changes to determine that an independent reviewer evaluated and approved change requests via a GitHub pull request.	No exceptions noted.

Control Objective 6: Change Management

Controls provide reasonable assurance that GitHub identifies the need for changes to the system, makes the changes following a controlled change management process, and prevents unauthorized changes from being made.

	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Service Auditor's Test	Results of Test
6.4	Technical reviews are performed for new features and major changes to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production.	Inspected tickets for a sample of new features and major changes to determine that technical reviews were performed to assess security, data, and architecture risk and that cross functional approval was required prior to release of changes to production.	No exceptions noted.
6.5	GitHub engineering code owners create and execute automated regression test cases to address functionality and security requirements.	Inspected a sample of pull requests to determine that GitHub engineering code owners created and executed automated regression test cases to address functionality and security requirements.	No exceptions noted.
6.6	Application changes pass automated continuous integration testing prior to deployment to the production environment.	Inspected the continuous integration testing configuration to determine that application changes were required to pass automated continuous integration testing prior to deployment to the production environment.	No exceptions noted.
6.7	A security code analysis tool scans code during commits and merges. The scanning tool posts potential vulnerabilities on the change pull request. The developer assesses the potential vulnerabilities, and a peer reviewer follows up on the items during change review and approval.	Inspected a sample of pull requests to determine that the security code analysis tool scanned the code during the commit and merge, the scanning tool posted potential vulnerabilities, the developer assessed potential vulnerabilities, and the peer reviewer followed up on items during change review and approval.	No exceptions noted.
6.8	Test environments are logically separated from production environments.	Inspected the test and production environments to determine that they were logically separated from each other.	No exceptions noted.

Control Objective 6: Change Management *Controls provide reasonable assurance that GitHub identifies the need for changes to the system, makes the changes following a controlled change management process, and prevents unauthorized changes from being made.*			
	Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	**Service Auditor's Test**	**Results of Test**
6.9	Branch protection system configurations enforce peer reviews and integration tests prior to the deployment of changes to the production environment and alert code owners of any forced deployments.	Inspected branch protection system configurations and an example change to the production environment to determine that branch protection system configurations enforced peer reviews and integration tests prior to the deployment of changes to the production environment and alerted code owners of any forced deployments.	No exceptions noted.
6.10	The Production Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems.	Inspected real-time monitoring of emergency deployments by the Production Engineering team to determine that the team monitored emergency deployments on a real-time basis.	No exceptions noted.
		Inspected documentation of a sample of emergency changes that impacted production systems to determine that a post-mortem review was performed.	No exceptions noted.
6.11	Infrastructure supporting the service is patched weekly as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.	Inspected evidence of patching for a sample of in-scope infrastructure components to determine that infrastructure supporting the service was patched weekly as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service were hardened against security threats.	No exceptions noted.

Section 5

Other Information Provided by GitHub, Inc.

Management's Response to Testing Exceptions

Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Results of Tests	Management's Response
Hubbers and contractors complete security and privacy awareness training within 45 days of their start date and Hubbers complete security and privacy awareness training on an annual basis.	Exceptions noted. 3 out of a sample of 19 new Hubbers did not complete security and privacy awareness training within the stipulated deadline. However, all new contractors out of a sample of 6 new contractors did complete security and privacy awareness training within the stipulated deadline.	GitHub is committed to continuing the improvement of our security and privacy training awareness program. GitHub has and will continue to make strides to drive timeliness in training completion through our monthly fundamentals program where leadership from across the company gathers to discuss company priorities, which includes training and resolving edge cases.
System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.	Exceptions noted. 2 out of a sample of 44 vulnerabilities identified during the sampled monthly internal and external vulnerability scans that were unable to be remediated within the required SLA were not handled via the documented exception process.	GitHub has a strong vulnerability management program and is committed to remediating vulnerabilities in accordance with documented SLAs and exception processes. When vulnerabilities fall outside of SLAs, they are escalated to management via the fundamentals process and corrective action is taken. GitHub continues to take action to eliminate vulnerabilities not remediated in accordance with documented SLAs and exception processes. The vulnerabilities noted in this exception have subsequently been remediated.

Management's Description of the Service Organization's Controls to Achieve the Related Control Objective	Results of Tests	Management's Response
The Security Operations team revokes logical production access, and Production Engineering revokes physical production access for terminated personnel (employee and contractors) within 24 hours of termination.	Exceptions noted. 2 out of a sample of 25 terminated employees and contractors did not have logical and physical access revoked within 24 hours of termination.	The root cause of the two overdue offboardings was an edge case related to the failure of the automated offboarding during a holiday break. Once the late offboardings were detected, the automation error was remediated and the affected offboardings were successfully processed promptly. GitHub performed investigative analysis and confirmed the departed personnel did not utilize or access any resources during the gap period. To mitigate against this in the future, monitoring alerts will be created and page the on call engineer for offboarding automation system failures.