*Proprietary & Confidential*

# GitHub
## Enterprise Cloud

**SOC 2**
Report on GitHub's System and Organization Controls
Relevant to Security

*Integrated SOC 2 Type 2 Report Prepared in Accordance with the AICPA Attestation Standards and IAASB ISAE No. 3000 (Revised) Standards*

OCTOBER 1, 2021 TO SEPTEMBER 30, 2022

MOSSADAMS

# Table of Contents

# I. Independent Service Auditor's Report

GitHub, Inc.
88 Colin P. Kelly Jr. St.
San Francisco, CA 94107

To the Management of GitHub:

## Scope

We have examined GitHub's accompanying description of its system in Section III titled "GitHub's Description of Its Enterprise Cloud" throughout the period October 1, 2021 to September 30, 2022 (description) based on the criteria for a description of a service organization's system in DC Section 200, *2018 Description Criteria for a Description of a Service Organization's System in a SOC 2® Report* (AICPA, *Description Criteria*) (description criteria) and the suitability of the design and operating effectiveness of controls stated in the description throughout the period October 1, 2021 to September 30, 2022, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the trust services criteria for Security (applicable trust services criteria) set forth in TSP Section 100, *2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy* (AICPA, *Trust Services Criteria*).

The information included in Section V titled "Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report" is presented by GitHub management to provide additional information and is not a part of GitHub's description. Information about GitHub's management responses to identified testing exceptions has not been subjected to the procedures applied in the examination of the description and of the suitability of the design and operating effectiveness of the controls to achieve GitHub's service commitments and system requirements based on the applicable trust services criteria, and accordingly, we express no opinion on it.

GitHub uses the following subservice organizations:

- Sabey for colocation data center services
- QTS for colocation data center services
- CoreSite for colocation data center services
- Equinix for colocation data center services
- Amazon Web Services for infrastructure hosting
- Microsoft Azure for infrastructure hosting

The description indicates that complementary subservice organization controls that are suitably designed and operating effectively are necessary, along with controls at GitHub, to achieve GitHub's service commitments and system requirements based on the applicable trust services criteria. The description presents GitHub's controls, the applicable trust services criteria, and the types of complementary subservice organization controls assumed in the design of GitHub's controls. The description does not disclose the actual controls at the subservice organizations. Our examination did not include the services provided by the subservice organizations, and we have not evaluated the suitability of the design or operating effectiveness of such complementary subservice organization controls.

The description indicates that complementary user entity controls that are suitably designed and operating effectively are necessary, along with controls at GitHub, to achieve GitHub's service commitments and system requirements based on the applicable trust services criteria. The description presents GitHub's controls, the applicable trust services criteria, and the complementary user entity controls assumed in the design of GitHub's controls. Our examination did not include such complementary user entity controls and we have not evaluated the suitability of the design or operating effectiveness of such controls.

## Service Organization's Responsibilities

GitHub is responsible for its service commitments and system requirements and for designing, implementing, and operating effective controls within the system to provide reasonable assurance that GitHub's service commitments and system requirements were achieved. GitHub has provided the accompanying assertion in Section II titled "GitHub's Assertion" (assertion) about the description and the suitability of design and operating effectiveness of controls stated therein. GitHub is also 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; selecting the applicable trust services criteria and stating the related controls in the description; and identifying the risks that threaten the achievement of the service organization's service commitments and system requirements.

## Service Auditor's Responsibilities

Our responsibility is to express an opinion on the description and on the suitability of the design and operating effectiveness of the controls 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 (AICPA) and in accordance with International Standard on Assurance Engagements (ISAE) 3000 (Revised), *Assurance Engagements Other Than Audits or Reviews of Historical Financial Information*, issued by the International Auditing and Assurance Standards Board (IAASB). Those standards require that we plan and perform our examination to obtain reasonable assurance about whether, in all material respects, the description is presented in accordance with the description criteria and the controls stated therein were suitably designed and operated effectively to provide reasonable assurance that the service organization's service commitments and system requirements were achieved based on the applicable trust services criteria. We believe that the evidence we obtained is sufficient and appropriate to provide a reasonable basis for our opinion.

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

- Obtaining an understanding of the system and GitHub's service commitments and system requirements
- Assessing the risks that the description is not presented in accordance with the description criteria and that controls were not suitably designed or did not operate effectively
- Performing procedures to obtain evidence about whether the description is presented in accordance with the description criteria
- Performing procedures to obtain evidence about whether the controls stated in the description were suitably designed to provide reasonable assurance that the service organization achieved its service commitments and system requirements based on the applicable trust services criteria

- Testing the operating effectiveness of the controls stated in the description to provide reasonable assurance that the service organization achieved its service commitments and system requirements based on the applicable trust services criteria
- Evaluating the overall presentation of the description

Our examination also included performing such other procedures as we considered necessary in the circumstances.

## Service Auditor's Independence and Quality Control

We are required to be independent and to meet our other ethical responsibilities in accordance with the Code of Professional Conduct established by the AICPA and the International Ethics Standards Board for Accountants' Code of Ethics for Professional Accountants.

We applied the Statements on Quality Control Standards established by the AICPA 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 report users and may not, therefore, include every aspect of the system that individual users may consider important to meet their informational needs.

There are inherent limitations in the effectiveness of any system of internal control, including the possibility of human error and the circumvention of controls.

Because of their nature, controls may not always operate effectively to provide reasonable assurance that the service organization's service commitments and system requirements are achieved based on the applicable trust services criteria. Also, the projection to the future of any conclusions about the suitability of the design and operating effectiveness of controls is subject to the risk that controls may become inadequate because of changes in conditions or that the degree of compliance with the policies or procedures may deteriorate.

## Description of Tests of Controls

The specific controls we tested and the nature, timing, and results of our tests are listed in Section IV of this report titled "Trust Services Category, Criteria, Related Controls, and Tests of Controls."

## Opinion

In our opinion, in all material respects:

- the description presents GitHub's Enterprise Cloud that was designed and implemented throughout the period October 1, 2021 to September 30, 2022, in accordance with the description criteria.
- the controls stated in the description were suitably designed throughout the period October 1, 2021 to September 30, 2022 to provide reasonable assurance that GitHub's service commitments and system requirements would be achieved based on the applicable trust services criteria, if the controls operated effectively throughout that period, and if the subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout that period.

- the controls stated in the description operated effectively throughout the period October 1, 2021 to September 30, 2022 to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria, if complementary subservice organization controls and complementary user entity controls assumed in the design of GitHub's controls operated effectively throughout that period.

## Restricted Use

This report, including the description of tests of controls and results thereof in Section IV, is intended solely for the information and use of GitHub, user entities of GitHub's Enterprise Cloud during some or all of the period October 1, 2021 to September 30, 2022, business partners of GitHub subject to risks arising from interactions with the Enterprise Cloud, practitioners providing services to such user entities and business partners, prospective user entities and business partners, and regulators who have sufficient knowledge and understanding of the following:

- The nature of the service provided by the service organization
- How the service organization's system interacts with user entities, business partners, subservice organizations, and other parties
- Internal control and its limitations
- Complementary user entity controls and complementary subservice organization controls and how those controls interact with the controls at the service organization to achieve the service organization's service commitments and system requirements
- User entity responsibilities and how they may affect the user entity's ability to effectively use the service organization's services
- The applicable trust services criteria
- The risks that may threaten the achievement of the service organization's service commitments and system requirements and how controls address those risks

This report is not intended to be, and should not be, used by anyone other than these specified parties.

Moss Adams LLP

San Francisco, California
November 11, 2022

## II. GitHub’s Assertion

We have prepared the accompanying description of GitHub’s system in Section III titled “GitHub’s Description of Its Enterprise Cloud” throughout the period October 1, 2021 to September 30, 2022 (description) based on the criteria for a description of a service organization’s system in DC Section 200, *2018 Description Criteria for a Description of a Service Organization’s System in a SOC 2® Report* (AICPA, *Description Criteria*) (description criteria). The description is intended to provide report users with information about the Enterprise Cloud that may be useful when assessing the risks arising from interactions with GitHub’s Enterprise Cloud, particularly information about system controls that GitHub has designed, implemented, and operated to provide reasonable assurance that its service commitments and system requirements were achieved based on the trust services criteria relevant to Security (applicable trust services criteria) set forth in TSP Section 100, 2017 *Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy* (AICPA, *Trust Services Criteria*).

GitHub uses the following subservice organizations:

- Sabey for colocation data center services
- QTS for colocation data center services
- CoreSite for colocation data center services
- Equinix for colocation data center services
- Amazon Web Services for infrastructure hosting
- Microsoft Azure for infrastructure hosting

The description indicates that complementary subservice organization controls that are suitably designed and operating effectively are necessary, along with controls at GitHub, to achieve GitHub’s service commitments and system requirements based on the applicable trust services criteria. The description presents GitHub’s controls, the applicable trust services criteria, and the types of complementary subservice organization controls assumed in the design of GitHub’s controls. The description does not disclose the actual controls at the subservice organizations.

The description indicates that complementary user entity controls that are suitably designed and operating effectively are necessary, along with controls at GitHub, to achieve GitHub’s service commitments and system requirements based on the applicable trust services criteria. The description presents GitHub’s controls, the applicable trust services criteria, and the complementary user entity controls assumed in the design of GitHub’s controls.

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

- the description presents GitHub's Enterprise Cloud that was designed and implemented throughout the period October 1, 2021 to September 30, 2022, in accordance with the description criteria.
- the controls stated in the description were suitably designed throughout the period October 1, 2021 to September 30, 2022 to provide reasonable assurance that the GitHub service commitments and system requirements would be achieved based on the applicable trust services criteria, if the controls operated effectively throughout the period, and if the subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout that period.
- the controls stated in the description operated effectively throughout the period October 1, 2021 to September 30, 2022 to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria, if complementary subservice organization controls and complementary user entity controls assumed in the design of GitHub's controls operated effectively throughout that period.

# III. GitHub’s Description of Its Enterprise Cloud

## A. Services Provided

### COMPANY OVERVIEW

GitHub, an independently operated Microsoft subsidiary, generated its first commit in 2007. It’s headquartered in San Francisco, California, with additional offices in Bellevue, WA; Raleigh, NC; Oxford, UK; Tokyo, Japan; Hyderabad, India; and Amsterdam, Netherlands. GitHub currently employs approximately 3,000 employees, with approximately 95 percent of the workforce being remote.

### SYSTEM DESCRIPTION

GitHub is a web-based software development platform built on the Git version control software. Primarily used to develop and manage 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, collaboration, bug tracking, feature requests, task management, GitHub advanced security, GitHub actions, pull requests, discussions, issues, pages, projects, docs, and wikis.

GitHub’s Enterprise Cloud is GitHub’s SaaS solution for collaborative software development. Features of the Enterprise Cloud service include:

#### 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 (SVN) clients or working with large files using Git Large File Storage (LFS).

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, has evolved.

GitHub allows sharing where 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 to issues, and link directly to the latest download.

#### CODE MANAGEMENT

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

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

PRs 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 someone else's 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 contribute to better quality code management. Repositories can be configured to require status checks, such as continuous integration tests, reducing both human error and administrative overhead.

### GITHUB ACTIONS

GitHub Actions automates Continuous Integration/Continuous Delivery (CI/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 like 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 do things like 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.

### PROJECT MANAGEMENT

Project boards allow users to reference every issue and PR in a card, providing a drag-and-drop 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 assigned tasks within a team. Issues may be used to track a bug, discuss an idea with an @mention, or start distributing work. Issue and PR 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 PRs to organize and track progress on groups of issues or PRs in a repository.

### TEAM MANAGEMENT

Building software is as much about managing teams and communities as it is about code. Users can 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.

GitHub Teams organizes people, provides level-up access with administrative roles, and tunes permissions for nested teams. Discussion threads keep conversations on topic using moderation tools, like issue and PR 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 docs in the text formatting language of choice, such as Textile or GitHub Flavored Markdown.

#### SYSTEM BOUNDARIES

The scope of this report includes GitHub's Enterprise Cloud, and the supporting production systems, infrastructure, software, people, procedures, and data. The following Enterprise Cloud features are included in the scope of this report: Issues, Pull Requests (PRs), Discussions, Wikis, Pages, Projects, Docs, Audit Logging, Advanced Security, Dependabot, and GitHub Actions.

#### SUBSERVICE ORGANIZATIONS

GitHub uses multiple subservice organizations in conjunction with providing its Enterprise Cloud product. GitHub uses Sabey, QTS, CoreSite, and Equinix to provide colocation data center services, and 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 *Complementary Subservice Organization Controls.*

## B. Principal Service Commitments and System Requirements

GitHub designs its processes and procedures to provide a secure environment for customer data. GitHub's security commitments are documented and communicated to customers in the Terms of Service and at other resources listed below:

- Security at GitHub https://github.com/security
- GitHub Privacy Statement https://help.github.com/en/articles/github-privacy-statement
- Terms of Service https://help.github.com/en/articles/github-terms-of-service

## C. Components of the System Used to Provide the Services

### 1. Infrastructure

Infrastructure consists of the colocation data centers, networks, systems, and other hardware powering the Enterprise Cloud product. Critical components of Enterprise Cloud's infrastructure include:

- 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.

- Web front-end servers are the forward-facing Hypertext Transfer Protocol Secure (HTTPS) servers for Enterprise Cloud. These servers provide the feature set for Enterprise Cloud.
- Application servers are used for processing asynchronous jobs or back-end processes required to support the web front-end. This processing may include replication between physical data centers, sending webhooks to repository integrations, sending emails, or performing other back-end processing.
- Email servers send email notifications to users or receive email issue comments from users.
- Git proxy servers manage the Git interaction between the users and the GitHub file servers.
- Advanced Programming Interface (API) front-end servers are the interface to any client using the API to interact with GitHub programmatically.
- Virtual Private Network (VPN) servers are used by GitHub employees to create a secure channel and an initial layer of authorization for employees who are developing or maintaining Enterprise Cloud.
- Bastion hosts, also referred to as jump hosts, are used by GitHub employees to manage the Enterprise Cloud environment.
- Database servers store the issues, milestones, and other project management information.
- Git Infrastructure File Servers are where the code is stored for Git code repositories.
- Actions Premium Runners execute customer's Actions jobs and are hosted in Azure.

## DATA CENTERS

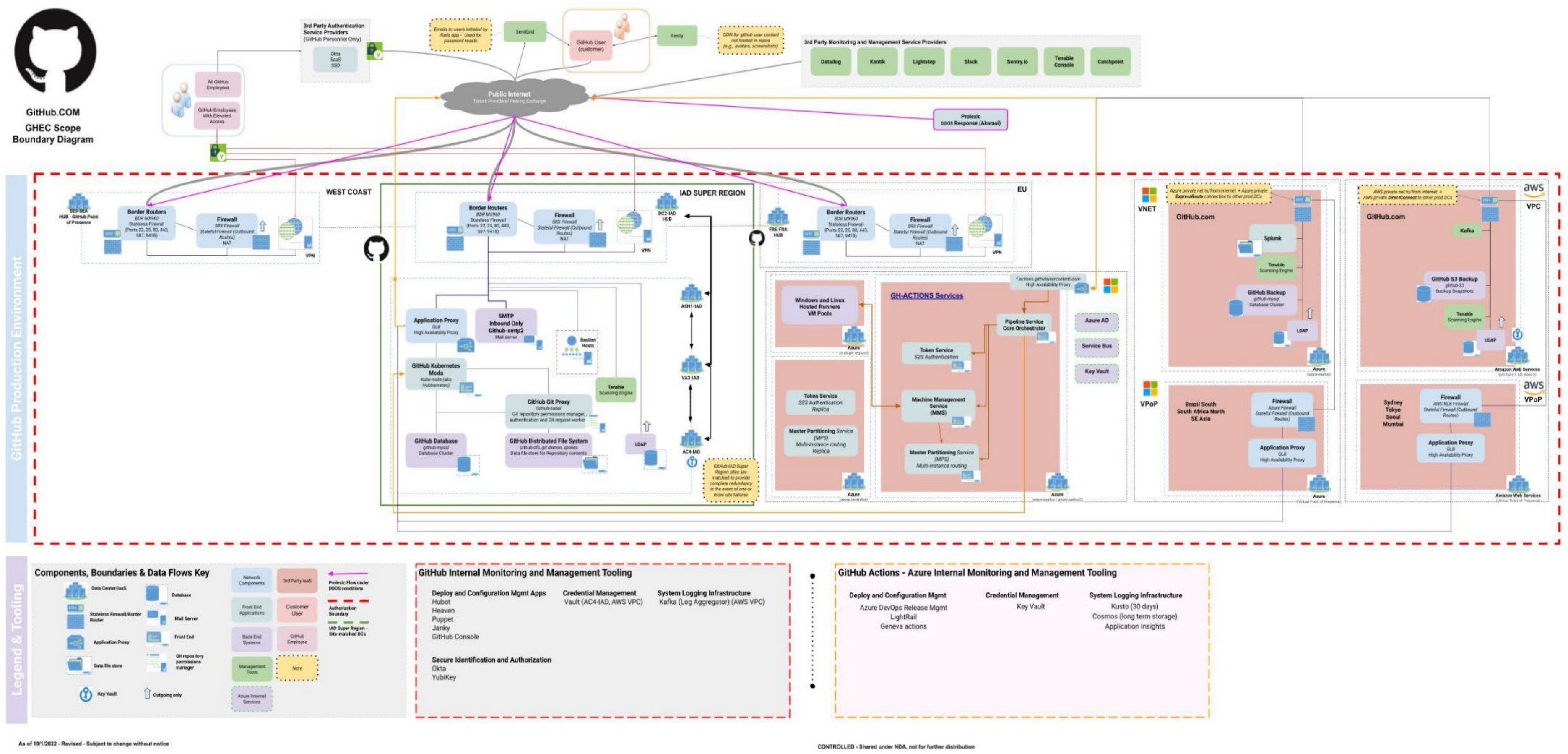
Enterprise Cloud infrastructure is hosted in geographically distributed data centers in Virginia, with virtual points-of-presence throughout the world.

Backups of customer Git repositories are performed on a real-time basis using an automated system. Backups of customer databases are encrypted during creation. Separately, backups from the Git file servers are maintained in geographically distinct AWS Simple Storage Solution (S3) data center locations. Backups from database servers are maintained in geographically distinct Azure Blob Storage locations.

Data center media used to store production data are destroyed onsite. Certificates verifying media destruction are documented and retained.

## NETWORK ARCHITECTURE

GitHub employs a demilitarized zone (DMZ), where application, database, and file servers are located. The system inside the DMZ is implemented as a multi-tier architecture. Application traffic flows in from the internet to GitHub's back-end infrastructure through border routers functioning as stateless firewalls. GitHub has a number of transit providers which land within its infrastructure on border routers. These routers provide scalable routing and stateless filtering services before packets enter GitHub's DMZ.


GitHub.COM
GHEC Scope
Boundary Diagram
3rd Party Authentication Service Providers (GitHub Personnel Only)
Okta SaaS SSO
All GitHub Employees
GitHub Employees With Elevated Access
Emails to users initiated by Rails app - Used for password resets
SendGrid
GitHub User (customer)
Fastly
CDN for github user content not hosted in repos (e.g., avatars, screenshots)
3rd Party Monitoring and Management Service Providers
Datadog
Kentik
Lightstep
Slack
Sentry.io
Tenable Console
Catchpoint
Public Internet
Transit Providers/ Peering Exchange
Prolexic DDOS Response (Akamai)
GitHub Production Environment
WEST COAST
SE3-SEA HUB - GitHub Point of Presence
Border Routers BDR MX960 Stateless Firewall (Ports 22, 25, 80, 443, 587, 9418)
Firewall SRX Firewall Stateful Firewall (Outbound Routes) NAT
VPN
IAD SUPER REGION
DC2-IAD HUB
ASH1-IAD
VA3-IAD
AC4-IAD
Application Proxy GLB High Availability Proxy
SMTP Inbound Only Github-smtp2 Mail server
Bastion Hosts
GitHub Kubernetes Moda Kube-node (aka Hubbernetes)
Tenable Scanning Engine
GitHub Git Proxy Github-babel Git repository permissions manager, authentication and Git request worker
GitHub Database github-mysql Database Cluster
GitHub Distributed File System Github-dfs, git demon, spokes Data file store for Repository contents
LDAP
GitHub IAD Super Region sites are matched to provide complete redundancy in the event of one or more site failures
EU
FR5-FRA HUB
Windows and Linux Hosted Runners VM Pools
Azure (multiple regions)
GH-ACTIONS Services
*.actions.githubusercontent.com High Availability Proxy
Pipeline Service Core Orchestrator
Token Service S2S Authentication
Machine Management Service (MMS)
Master Partitioning Service (MPS) Multi-instance routing
Token Service S2S Authentication Replica
Master Partitioning Service (MPS) Multi-instance routing Replica
Azure (azure-centralus)
Azure (azure-eastus / azure-eastus2)
Azure AD
Service Bus
Key Vault
VNET
Azure private net to/from internet + Azure private ExpressRoute connection to other prod DCs
GitHub.com
Splunk
Tenable Scanning Engine
GitHub Backup github-mysql Database Cluster
LDAP
Azure (azure-eastus)
VPoP
Brazil South South Africa North SE Asia
Firewall Azure Firewall Stateful Firewall (Outbound Routes)
Application Proxy GLB High Availability Proxy
Azure (Virtual Point of Presence)
aws
VPC
AWS private net to/from internet + AWS private DirectConnect to other prod DCs
GitHub.com
Kafka
GitHub S3 Backup github-S3 Backup Snapshots
Tenable Scanning Engine
LDAP
Amazon Web Services (US East 1, US West 2)
Sydney Tokyo Seoul Mumbai
Firewall AWS NLB Firewall Stateful Firewall (Outbound Routes)
Application Proxy GLB High Availability Proxy
Amazon Web Services (Virtual Point of Presence)
Legend & Tooling
Components, Boundaries & Data Flows Key
Data Center/IaaS
Database
Stateless Firewall/Border Router
Mail Server
Application Proxy
Front End
Data file store
Git repository permissions manager
Key Vault
Outgoing only
Network Components
3rd Party IaaS
Front End Applications
Customer User
Back End Systems
GitHub Employee
Management Tools
Note
Azure Internal Services
Prolexic Flow under DDOS conditions
Authorization Boundary
IAD Super Region - Site-matched DCs

GitHub Internal Monitoring and Management Tooling

**Deploy and Configuration Mgmt Apps**
Hubot
Heaven
Puppet
Janky
GitHub Console

**Credential Management**
Vault (AC4-IAD, AWS VPC)

**System Logging Infrastructure**
Kafka (Log Aggregator) (AWS VPC)

**Secure Identification and Authorization**
Okta
YubiKey

GitHub Actions - Azure Internal Monitoring and Management Tooling

**Deploy and Configuration Mgmt**
Azure DevOps Release Mgmt
LightRail
Geneva actions

**Credential Management**
Key Vault

**System Logging Infrastructure**
Kusto (30 days)
Cosmos (long term storage)
Application Insights

As of 10/1/2022 - Revised - Subject to change without notice

CONTROLLED - Shared under NDA, not for further distribution

Filtering is configured to only accept traffic from known routes, enforce ingress and egress routing policy, and implement port-based access control lists (ACLs). Once packets traverse the border infrastructure and enter the DMZ, they are passed to the application-layer proxies, which are responsible for associating the packet with a service.

## 2. Software

The software consists of the system software that supports application programs (operating systems, middleware, and utilities) for the Enterprise Cloud product. GitHub's software stack consists of Linux servers running Nginx, Unicorn, and MySQL databases. Datastores such as Redis, Memcached, Elasticsearch, and others are also used to support the primary environment.

Most user-visible product features on Enterprise Cloud, as well as the GitHub API, are maintained under a single Ruby on Rails 5 application. Supporting services and applications are written primarily in Golang, C, and NodeJS.

Linux servers run on Debian Stretch or Jessie, with server build configurations generated from data in Puppet, GitHub's configuration management tool. The hardware is managed by gPanel, an internally developed hardware management platform. When a new device is detected, it is forced to Pre eXecution Environment (PXE) network boot to receive the GitHub image. Then, depending on the function that hardware will perform, it is bootstrapped with the latest version of the correct software.

## 3. People

The personnel primarily involved in the security, governance, operation, and management of GitHub include the following:

- *Senior Leadership* - Responsible for the overall governance of GitHub. This group includes the CEO, Chief Financial Officer (CFO), Chief Revenue Officer (CRO), Chief Human Resource Officer (CHRO), Head of Design, Vice President of Strategy, Chief Security Officer (CSO) and Senior Vice President of Engineering, Chief Legal Officer, and Vice President of Communities.
- *Security* - Responsible for ensuring the confidentiality, integrity, availability, and privacy of data handled by GitHub is protected, and strategic product and operational initiatives have secure design in systems, applications, and processes. Security consists of multiple teams with specific missions: Product Security Engineering, Threat Hunting, Operations, and Response (THOR), Security Incident Response Team (SIRT), Security Lab, Security Operations, Secure Access Engineering, Security Telemetry, Vulnerability Management, Cloud and Enterprise Security, IT and Business Systems, Data Protection, and Governance, Risk, Compliance, and Communication (GRCC). These teams manage security incident detection and response, monitoring, vulnerability scanning, network and application layer penetration testing, secure architecture, secure engineering and operations, access management, endpoint asset management, and risk and compliance oversight.
- *Product & Application Engineering* - Responsible for understanding customer requirements, collecting, defining, and clarifying feature requests and development efforts, and managing feature rollouts and related customer communication efforts. Developers on the Application Engineering team work with the Product team to plan and coordinate releases, and they are accountable for building, testing, and deploying Enterprise Cloud code and feature changes.

- *IT* - Responsible for managing corporate IT services and support functions within GitHub, including endpoint security maintenance, deskside and remote support, managing procurement and distribution of laptops, software licenses, and other gear required by personnel, as well as fielding requests. Additionally, IT supports Security and Human Resources with employee and contractor onboarding and offboarding responsibilities.
- *Infrastructure* - Responsible for maintaining service availability, including performance and scale monitoring and reporting, incident command, and on-call readiness for any production issues. Infrastructure consists of multiple teams focused on additional aspects of production operations, including configuration management, building, testing, and deploying software relevant to the operation and management of production assets, patching and remediation of vulnerabilities reported by the Security team, and data center operations management. The Storage Engineering team within Platform manages Git and database storage backups and restores.
- *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.
- *Legal* - Responsible for negotiating contractual obligations with third parties and technology partners/suppliers, legal terms and conditions, ensuring compliance with internal contractual standards.
- *Privacy* - Responsible for determining which privacy laws and regulations apply to GitHub and determine the best way to comply with them, ultimately ensuring we can offer our products to every developer anywhere in the world.
- *Customer Security* - Responsible for providing technical and account-related support to Enterprise Cloud customers and for resolving customer issues via email, chat, social media, and phone from developers and customer entities around the globe.

## 4. Data

GitHub uses repository data to connect users to relevant tools, people, projects, and information. Repositories are categorized as either 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 has 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 selection.

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 will perform 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. Private data exchanged with GitHub is transmitted over Transport Layer Security (TLS). Send and receipt of private data is done over Secure Shell (SSH) authenticated with keys, or over HTTPS, using a GitHub username and password.

For more information about GitHub's use of customer data, refer to the following link: How GitHub Uses & Protects Your Data (https://docs.github.com/en/get-started/privacy-on-github/about-githubs-use-of-your-data).

## 5. Processes and Procedures

GitHub maintains programmatic (automated) and manual procedures involved in the operation of the Enterprise Cloud product. These 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. These procedures are drafted in alignment with the overall information security 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 Policy outlines the security roles and responsibilities for the organization and expectations for employees, contractors, and third parties using GitHub systems or data. The CSO reviews and approves the ISMS Policy annually.

This overarching security policy is supported by a number of dependent security policies, standards, and procedures that address the security of systems, facilities, data, personnel, and processes. GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes.

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

| Policy | Implementing Standard(s) | Procedures |
|---|---|---|
| **GitHub Information Security Management System (ISMS)** | | |
| **GitHub ISMS Scope** | | |
| **GitHub ISMS Statement of Applicability (SOA)** | | |
| **Standards directly associated with the ISMS:** | | |
| | Chatops Command Security and Risk Standard | |
| | Controls Monitoring Standard: | |
| | | Controls Monitoring SOP |
| | Data Classification Standard | |
| | Domain Management Standard | |
| | Endpoint Security Standard | |

| Policy | Implementing Standard(s) | Procedures |
|---|---|---|
| | Enterprise Administration Standard | |
| | External File Sharing Standard | |
| | Git Systems Server Site Failure plan | |
| | Heroku Standard | |
| | 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** | | |
| **Full Time Employee Termination Policy** | | |
| **Identity and Access Management Policy:** | | |
| | Identity and Access Management Standard: | |
| | | IAM Onboarding SOP |

| Policy | Implementing Standard(s) | Procedures |
|---|---|---|
| | | 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** | | |
| **Private Information Removal Policy - External Customer Facing Policy** | | |
| **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:** | | |

| Policy | Implementing Standard(s) | Procedures |
|---|---|---|
| | 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** | | |
| **Security Risk Management Policy:** | | |
| | | Security Risk Reporting - Standard Operating Procedure |
| | | Security-GRCC Vendor Risk Assessment Process |
| **System and Services Acquisition Policy:** | | |
| | Vendor Security Standard: | |
| | | Purchasing Workflow |
| | | Vendor Security Reviews SOP |
| | | Procurement Workflow |
| | | Vendor Off-boarding Checklist |
| | | Decommissioning a GitHub-Owned App |

| Policy | Implementing Standard(s) | Procedures |
|---|---|---|
| | | Vendor Offboarding SOP |
| | Encryption Standard | |
| **Vulnerability Management Policy:** | | |
| | Container Hardening Standard: | |
| | | Exception Handling Process |
| | | Vulnerability Management Process |
| | | Checklist for Docker Baseline Security |
| | Database Hardening Standard | |
| | OS Hardening Standard | |
| | Patch Management Standard | |
| | FedRAMP Vulnerability Reporting Standard: | |
| | | FedRAMP Annual Vulnerability Exception Review |
| | | FedRAMP Monthly Vulnerability Management Reporting Procedure |
| **Background Checks Policy** | | |
| **IT Asset Management Policy** | | |
| **Network Policy** | | |
| **Resilience Program Policy:** | | |
| | Resiliency Standard | |
| **Security Document Management Policy:** | | |
| | Security Document Management Standard | |

## D. Relevant Aspects of the Control Environment, Risk Assessment Process, Information and Communication, and Monitoring

### 1. Control Environment

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 the company'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.

#### CODE OF CONDUCT

During the onboarding process, new employees complete security and privacy awareness training and review and acknowledge the employee guide to company policies and practices, the Hubber Handbook. The Handbook includes the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics.

#### 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 also reports and reviews progress towards meeting those objectives semiannually.

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

GitHub's organizational structure is designed to meet the company's security requirements and commitments to customers. GitHub's Senior Leadership team, specifically those reporting to the Chief Executive Officer (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 where there may be a compliance impact. New incoming leadership is announced to the organization on the company intranet, and in regular all-hands meetings.

The Security team, headed by the CSO and SVP Engineering, is responsible for monitoring and enhancing GitHub's overall security posture. This function includes managing security risks and threats, educating employees on security-related best 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 across the company is accountable for ensuring necessary policy, standards, and standard operating procedures 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, strategic business initiatives driven by leadership, and other events that dictate changes. Security leadership reviews and approves the materials and any changes on an annual basis.

The Security team administers security and privacy awareness training on an annual basis. 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.

## HUMAN RESOURCES

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.

In order to be employed by GitHub, candidates must successfully complete a background check. The background check is initiated after the candidate signs the offer letter. Employees 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 VP 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 to 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 three-day company onboarding 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 their first week, new employees are introduced to the roles of Security, GRCC, and Data Protection in GitHub and are required to complete the security and privacy awareness training.

Additionally, product engineers hired to work on the Enterprise Cloud application and services complete Secure Coding Practices training. This training includes existing engineering employees re-taking the training when a refresh is required on coding best practices to help ensure engineers align with changes in the code base or development tools.

Furthermore, GitHub management and peers evaluate and provide feedback to employees annually.

## USER AUTHENTICATION

This section outlines the process for both internal and customer users to access the Enterprise Cloud.

### GITHUB INTERNAL ACCESS

GitHub has implemented access protection measures to only allow authenticated and authorized users access to the portions of GitHub's instance of the Enterprise Cloud instance that are not explicitly public. Enterprise Cloud requires a valid and unique account ID, password, and two-factor authentication (2FA). GitHub integrates with a third-party, single sign-on (SSO) provider that enforces two-factor authentication using FIDO2 authenticators.

Access to internal systems is restricted through unique account IDs ("handles"), password, and 2FA. To access these environments, GitHub personnel leverage the same GitHub handles as GitHub Enterprise Cloud. 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 Enterprise Cloud production systems is restricted to authorized personnel with demonstrated need and isolated from the internet by VPN, bastion hosts, and/or SAML enforcing 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 Security Access Engineering team has the ultimate responsibility for maintaining the Entitlements system, but the access grants and review flow are the responsibility of the user(s) requesting access and their manager.

### GITHUB CUSTOMER ACCESS

Access to customer organizations requires a unique account and password. Customers can optionally enable 2FA for their organization through Enterprise Cloud settings. Audit logging is also available to allow organization administrators to quickly review user actions performed. 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 is designed to limit access to data and settings to the user who created the repository or who has 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, 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 a number of 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: VPN, bastion host, and SAML enforcing HTTPS reverse proxy.

Each of these remote access mechanisms require multiple factors for authentication and employ 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 network 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 security information and event management (SIEM) tool. Alerting is configured to detect unusual or unauthorized activity and, when triggered, response is executed in accordance with the standard procedures defined within the SIRT's monitoring, detection, and response program. As needed, based on the frequency and type of activity detected, the SIRT identifies process improvements to reduce human interactions directly with production servers.

## ENCRYPTION

Production traffic is encrypted in transit over the public internet. Transmissions via SSL/TLS require a minimum of 2048-bit certificate keys, and GitHub uses TLSv1.2 and above to encrypt data during transit. Private data is transmitted via SSH or TLS. Customer Git repositories are encrypted at rest. Full disk encryption is required on company laptops and is enforced through a centralized endpoint management system.

## USER PROVISIONING

New employee permissions are automatically 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 Security Operations 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 reviews and approves the access request.

During onboarding, GitHub IT shepherds employees through configuring their account setup and 2FA, for GitHub's SSO provider, and GitHub organization. IT monitors the access configuration, and users are locked out of internal resources until remediated if they are found to not have required security settings enabled in accordance with company requirements.

The Security Operations team is responsible for facilitating a periodic review over access to sensitive systems. Security Operations identifies systems and elevated access that warrant additional review and oversight, given the potential impact on the Enterprise Cloud platform. Specifically, these systems include production, security management tools, user access tools, and vulnerability management tools. Managers review and approve access to critical production and access gateway systems for their direct reports on a semiannual basis. The reviews allow the manager to provide attestation that the level of access currently granted to each individual under their review is appropriate and required for the individual's job function. User access levels remaining after changes from the review are authorized by management, or removed if approval is not granted.

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

THOR builds automated monitors of user access events to identify anomalous user access activity. SIRT investigates detections, and if SIRT concludes 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. After People Operations enters an employee's termination date in GitHub's HRIS, an offboarding notification is sent to Security Operations daily, for both planned and unplanned exits. The Security Operations team works through a standard offboarding procedure, with most aspects of account termination being automated via the Entitlements system, with manual removal of user access when necessary.

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

## ANTI-VIRUS

Threat detection software is installed on GitHub employee endpoints to detect malicious software, with installation enforcement maintained through a centralized endpoint management tool. The collected data flows through a managed detection platform. The monitoring tool collects a variety of metadata about endpoint activity, which is provided to the detection platform via a normalized and aggregated data feed. This data is used to detect and verify anomalous activity on GitHub endpoints. Data regarding suspicious activity detected is escalated to GitHub's SIRT for further investigation and response.

## CONFIGURATION MANAGEMENT

System hardening standards have been documented, based on GitHub security practices. The Security Operations team reviews and approves changes to the standard hardening procedures.

Hardened system configurations are maintained in GitHub source code repositories and are deployed to production through a configuration management tool. This configuration management tool monitors system configurations and automatically reverts out-of-compliance systems to the approved hardening baseline.

Changes to the GitHub configurations are peer reviewed and approved prior to deploying the changes to production. Configuration change activities are logged in the GitHub's Puppet repository.

## VULNERABILITY MANAGEMENT

The GitHub Vulnerability Management Program monitors and interrogates public-facing and internal infrastructure to identify systems that are vulnerable to known exploits, configured in ways that unnecessarily increase risk of compromise, or have software installed which is known to be insecure. The program develops and maintains scanning coverage across all hosting platforms, providers, and networks. Vulnerability scanning is executed on at least a monthly basis, 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 SLA are handled via the documented exception process.

GitHub Product Security Engineering, Vulnerability Management, 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, 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, developing and maintaining 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 non-disclosure agreements (NDAs).

### SYSTEM MONITORING

THOR actively instruments 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 event(s) it is designed to detect and surface. SLAs are as follows:

| Severity | SLA | Intended Response | Example |
|---|---|---|---|
| **Critical** | 30 minutes – 24/7 | Immediate human response | Lost device |
| **High** | 24 hours | Human response within one day | High confidence IOC match or behavioral detection |
| **Medium** | 1-7 days | Automated or human response within a week | Low confidence IOC match |
| **Low** | N/A | Automated response or contextual 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 post-mortem conclusions in the relevant security repository issue for that event and resolves the issue when complete.

### INCIDENT RESPONSE

The THOR and SIRT teams triage, investigate, and respond to a variety of security events reported by internal and external sources, including:

- Suspected internal security compromises
- Critical dotcom vulnerabilities or bugs that may expose user or customer data
- GitHub or customer credential/secret exposure
- External requests for information on dotcom contents or fetches/page views in support of user or customer security incidents

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

Initial reports are defined as "leads" until appropriately triaged by a member of the relevant THOR or SIRT team member. Once either THOR or SIRT verifies a lead, an escalation procedure is executed to engage appropriate resources to help ensure 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 varies 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.

Incident severities are classified as follows; SLAs represent the time from the moment the lead is validated to mitigation of the root cause:

| Severity | SLA | Potential Result |
|---|---|---|
| **Critical** | 7 days | Catastrophic negative impact to the reputation of GitHub, GitHub personnel (Hubbers), users, or customers<br>Catastrophic sales losses and/or fiscal penalties associated with breach |
| **High** | 14 days | Substantial negative impact to the reputation of GitHub, Hubbers, users, or customers<br>Substantial sales losses and/or fiscal penalties associated with breach |
| **Medium** | 30 days | Moderate negative impact to the reputation of GitHub, Hubbers, users, or customers<br>Moderate sales losses |
| **Low** | 60 days | Limited to no negative impact to the reputation of GitHub, Hubbers, users, or customers<br>No sales losses |

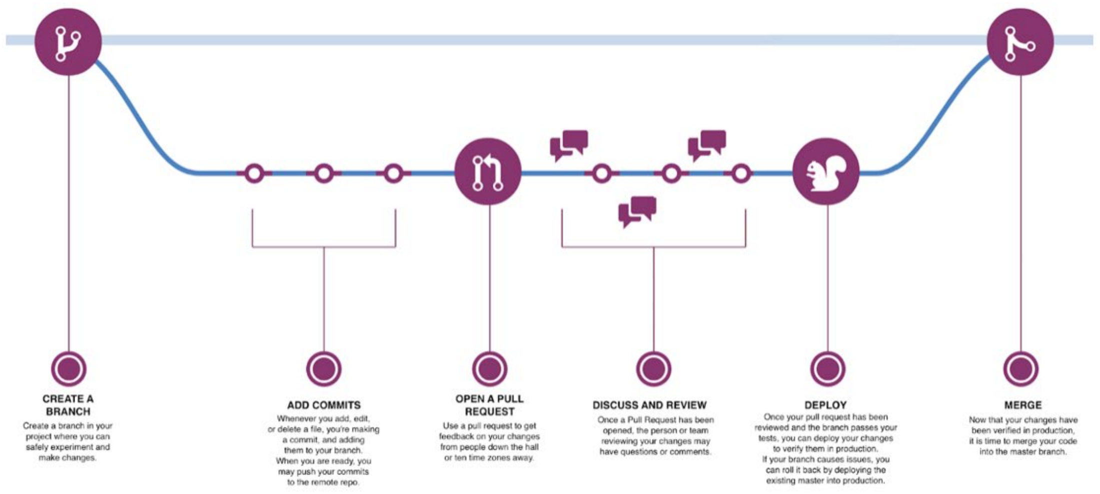
The THOR and SIRT teams also document Legal engagement and external notification processes in the event an incident is validated as a breach or in violation of contractual commitments. GitHub informs affected users and organizations as required. The messaging can be sent via various forms such as email, 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

GitHub is a collaborative, organic, and adaptable organization. GitHub's code and system change management processes leverage Enterprise Cloud'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 Enterprise Cloud 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 Enterprise Cloud GitHub repositories is restricted to members of the Platform and Product Engineering teams. Engineers responsible for the critical EC Repositories of the codebase receive timely notifications of changes pushed and monitor these changes through automated issues posted in the Enterprise Cloud repositories and in Slack.



**CREATE A BRANCH**
Create a branch in your project where you can safely experiment and make changes.

**ADD COMMITS**
Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. When you are ready, you may push your commits to the remote repo.

**OPEN A PULL REQUEST**
Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

**DISCUSS AND REVIEW**
Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments.

**DEPLOY**
Once your pull request has been reviewed and the branch passes your tests, you can deploy your changes to verify them in production. If your branch causes issues, you can roll it back by deploying the existing master into production.

**MERGE**
Now that your changes have been verified in production, it is time to merge your code into the master branch.

When developing code for the Enterprise Cloud product, engineers create a branch from the Default Branch, which is the 'gold copy' of GitHub, to begin development of changes. This ensures that test environments are logically separated from production environments. Changes made on these branches are deployed to a pre-production environment for testing. Once pre-production testing has validated the change is functioning as expected, the branch is committed to the Default 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 Application Security 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 the systems. Engineers are trained on the requirements for performing an effective peer review, and each team overlays domain-specific training and checks to the review. The Security team conducts high-level design reviews, including security reviews, and release reviews for large projects and initiatives to assess security risk. For production changes, a code review and, where relevant, a review of database schema occurs to verify the integrity of the changes.

Once the code is ready for discussion and review, developers open a PR for commits added to their branch. PRs are an integral feature of Enterprise Cloud used to coordinate the discussion, approval, testing, and deployment of changes.

Enterprise Cloud repositories come with the ability to designate specific users as "Code Owners". Code Owners are responsible for reviewing and approving PRs to ensure the appropriateness of commits. Additionally, repository owners have the ability to create a "Code Owners File" that acts as a configuration for the repository contents, enforcing required reviews by specifically nominated users prior to deploying PRs to production and then as part of a merge action to the repository's Main branch. Specific users may be designated as mandatory reviewers for all or select portions of the repository's contents/code.

Protected branches (branches where merging to the Main branch is gaited by required review and approval activities) are also enabled on the Enterprise Cloud code repositories, requiring commits to branches via PRs to be reviewed by an independent reviewer prior to merging the code to the Default Branch.

### AUTOMATED TESTING

Continuous Integration (CI) is the process by which code changes are automatically tested and integrated into a single PR. When a change is committed to a branch with an associated PR, the CI system automatically executes the configured CI status checks for that repository. This process includes the following:

- 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 PR

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 being 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, which include regression test cases, 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.

Peer review is the primary method used by GitHub to help ensure the appropriate level of test cases have been implemented prior to deploying changes. Code Owners are domain experts over repositories and/or specific areas of the code. These individuals are ultimately responsible for helping to ensure tests are created and deprecated in accordance with the functional requirements.

## DEPLOYMENT

To support deployment to the production environment, changes are deployed using GitHub's standard and proprietary deployment tools and workflows. GitHub's automated deployment tool, Heaven, sits within GitHub's production environment to manage packages deployed to the production systems. To deploy any application or configuration changes, developers run a Slack Chat Operations (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. Furthermore, Heaven ensures required checks are completed, including passing of mandatory acceptance checks and ensuring the most recent Default 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 Enterprise Cloud product, comprehensive logging and monitoring tools are built into the Enterprise Cloud application, system, and network to monitor real-time telemetrics such as network traffic, successful logins and repository actions, exceptions and error messages from the application and systems, and 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 Default 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 Default 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 teams regularly host internal demos for the organization as a part of regular company and technology all-hands get togethers. These presentations are a way for technology 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, fixed, or any security fixes (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 GitHub blog posts and may also include social posts and product training-videos.

For new features and major changes, technical reviews are performed to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production.

#### EMERGENCY DEPLOYS

When an emergency or system failure means a deployment 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 deployment process, and knowing when it is justified, is a controlled and necessary component of deployment processes.

Engineers with write access to a repository and privileges to deploy to production have the ability to force deploy, which bypasses required CI and normal approval processes. Forced deployments are requested in the same pre-defined deployment Slack channels used for regular deployments, and the engineer requesting the deployment specifies the emergency justification to give context to the situation to observers and others who are monitoring these deployments.

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 Default Branch. Both of these paths require the normally required CI and peer review to complete, ensuring the same level of visibility and testing as any other change to the code base.

When an emergency deployment is executed, an alert is posted to the #incident-command Slack channel for visibility to the on-call team. A post-mortem remediation issue is automatically created in the github/availability repository. The issue is assigned to the engineer who deployed with a post-mortem checklist the assignee completes within 24 hours, giving justification for the deployment.

Post-mortem 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-deployment steps to help ensure no further issues require resolution.

## 2. Risk Assessment Process

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 Security GRCC team is embedded within the larger Security team. This team monitors 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 Enterprise Cloud 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 Security GRCC 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 Security GRCC team 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, Security GRCC 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 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, IT, Business Systems, Legal, and Privacy review this annual risk report.

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.

### VENDOR INVENTORY

A vendor inventory is generated each quarter as part of regulatory reporting for other compliance requirements. The Security GRCC team generates this inventory using automated tools along with an inventory of production components to help ensure various aspects of the system inventory are reviewed and updated on a quarterly basis.

## 3. 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 Hubber Handbook and published separately, outlining the responsibility of employees to report security and operational failures, incidents, system problems, and complaints. The document owner(s) and Security leadership review and approve security policies and procedures annually. Significant changes to policies result in communication to personnel regarding policy updates.

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.

### EXTERNAL COMMUNICATIONS

GitHub communicates the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints using its official blog (https://blog.github.com). 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://blog.github.com/changelog) and available for RSS feed subscription. It is also accessible via GitHub's Changelog twitter account (@GHChangelog).

Enterprise Cloud users have ready access to support resources at GitHub Help (https://help.github.com) and GitHub Guides (https://guides.github.com), and they can access customized feature tutorials through the GitHub Learning Labs (https://lab.github.com).

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

The user community can communicate directly with GitHub. GitHub 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.

## 4. Monitoring Controls

The Security GRCC team at GitHub is responsible for monitoring the internal control environment for each of the compliance frameworks adopted by GitHub.

### INTERNAL CONTROL REVIEWS

The Security GRCC 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 up to the Senior Leadership team for awareness on a quarterly basis.

## E. Trust Services Category, Criteria, and Related Controls

Although the applicable trust services category, criteria, and related controls are presented in Section IV of this report titled "Trust Services Category, Criteria, Related Controls, and Tests of Controls", they are an integral part of GitHub's system description throughout the period October 1, 2021 to September 30, 2022.

## F. Complementary Subservice Organization Controls

GitHub's controls related to the Enterprise Cloud cover only a portion of overall internal control for each user entity of GitHub. It is not feasible for the criteria related to the Enterprise Cloud to be achieved solely by GitHub. Therefore, each user entity's internal controls must be evaluated in conjunction with GitHub's controls and the related tests and results described in Section IV of this report, taking into account the types of controls expected to be implemented by the subservice organization as described below.

| | Complementary Subservice Organization Controls | Related Criteria |
|---|---|---|
| **AWS, Azure** | | |
| 1 | Access to the in-scope systems requires users to securely authenticate before being granted access. | CC6.1 |
| 2 | User content is segregated and made viewable only to authorized individuals. | CC6.1 |
| 3 | New user accounts are approved by appropriate individuals prior to being provisioned. | CC6.2 |
| 4 | User accounts are removed when access is no longer needed. | CC6.2 |
| 5 | User accounts are periodically reviewed to verify the accounts, and their permissions, are current and appropriate. | CC6.2 |
| 6 | Access modifications are approved by appropriate individuals prior to being provisioned. | CC6.3 |
| 7 | User accounts are removed when access is no longer needed. | CC6.3 |
| 8 | User accounts are periodically reviewed to verify the accounts, and their permissions, are current and appropriate. | CC6.3 |

| | Complementary Subservice Organization Controls | | Related Criteria |
|---|---|---|---|
| 9 | Only authorized users have access to the physical facilities securing the system. | ➤ | CC6.4 |
| 10 | Production media is securely decommissioned and physically destroyed prior to being removed from the data center. | ➤ | CC6.5 |
| 11 | Network security mechanisms restrict external access to the production environment. | ➤ | CC6.6 |
| 12 | Access to customer data is restricted to appropriate users. | ➤ | CC6.7 |
| 13 | Customer data is protected during transmission. | ➤ | CC6.7 |
| 14 | Anti-virus or anti-malware solutions are installed to detect or prevent unauthorized or malicious software. | ➤ | CC6.8 |
| 15 | Vulnerabilities are logged and tracked to resolution. | ➤ | CC7.1 |
| 16 | Security events on system components are monitored and evaluated to determine potential impact per policy. | ➤ | CC7.2 |
| 17 | Security events are reviewed to determine whether they should be elevated to an incident. | ➤ | CC7.3 |
| 18 | Security events deemed incidents are remediated and communicated. | ➤ | CC7.4 |
| 19 | System changes are documented, tested, and approved prior to migration to production. | ➤ | CC8.1 |
| **QTS, Sabey, CoreSite, Equinix** | | | |
| 1 | Only authorized users have access to the physical facilities securing the system. | ➤ | CC6.4 |
| 2 | Access to customer data is restricted to appropriate users. | ➤ | CC6.7 |
| 3 | Customer data is protected during transmission. | ➤ | CC6.7 |
| 4 | Security events on system components are monitored and evaluated to determine potential impact per policy. | ➤ | CC7.2 |
| 5 | Security events are reviewed to determine whether they should be elevated to an incident. | ➤ | CC7.3 |
| 6 | Security events deemed incidents are remediated and communicated. | ➤ | CC7.4 |
| 7 | System changes are documented, tested, and approved prior to migration to production. | ➤ | CC8.1 |

## G. Complementary User Entity Controls

GitHub's Enterprise Cloud was designed under the assumption that certain controls would be implemented by the user entities for whom it provides its Enterprise Cloud. In these situations, the application of specific controls at these user entities is necessary to provide reasonable assurance that the service organization's service commitments and system requirements were achieved based on the applicable trust services criteria.

This section describes additional controls that should be in operation at the user entities to complement the controls at GitHub. User auditors should consider whether the following controls have been placed in operation by the user entity.

Each user entity must evaluate its own internal control structure to determine if the identified user entity controls are in place. User entities are responsible for:

| | Complementary User Entity Controls | | Related Criteria |
|---|---|---|---|
| 1 | Enabling SAML for their Enterprise Cloud accounts. | ➤ | CC6.1 |
| 2 | Enabling two-factor authentication and ensuring members and collaborators require two-factor authentication. | ➤ | CC6.1, CC6.6 |
| 3 | Creating and managing their Organization and Teams, including the proper configuration of access permissions to repositories. | ➤ | CC6.2, CC6.3 |
| 4 | Inviting, removing, and managing users in their Organization and Teams, including granting of permission levels and access to repositories, and periodic review of Organization users and outside collaborators. | ➤ | CC6.2, CC6.3 |
| 5 | Ensuring authorized users are appointed as Organization owners for administration of the Organization. | ➤ | CC6.2, CC6.3 |
| 6 | Maintaining an effective onboarding and offboarding process for their own employees and contractors. | ➤ | CC6.2, CC6.3 |
| 7 | Reviewing events in the security logs. | ➤ | CC7.1 |
| 8 | Administering and configuring repositories, including permissions, enabling required reviews for pull requests, and enabling required status checks before merging. | ➤ | CC8.1 |
| 9 | Securing configuration of GitHub Actions. See https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions for detailed guidance. | ➤ | CC8.1 |
| 10 | Reviewing and authorizing third-party applications, properly configuring the GitHub Application Programming Interface (API), including connecting with third-party applications, and managing third-party application access to their own repositories and data, where applicable. | ➤ | CC9.2 |

# IV. Trust Services Category, Criteria, Related Controls, and Tests of Controls

This integrated SOC 2 Type 2 Report was prepared in accordance with the AICPA attestation standards and in accordance with the International Standard on Assurance Engagements (ISAE) 3000 (Revised), *Assurance Engagements Other Than Audits or Reviews of Historical Financial Information*, issued by the International Auditing and Assurance Standards Board (IAASB), and has been performed to examine the suitability of the design and operating effectiveness of controls to meet the criteria for the Security category set forth in TSP Section 100, *2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy* (AICPA, *Trust Services Criteria*) throughout the period October 1, 2021 to September 30, 2022.

The trust services category for the Security criteria and related controls specified by GitHub are presented in Section IV of this report.

## A. Applicable Trust Services Criteria

### Common Criteria

| CC1.0 | Common Criteria Related to Control Environment | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC1.1** | The entity demonstrates a commitment to integrity and ethical values. | 1 | 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. | No exceptions noted. |
| | | 9 | Individuals with access to the GitHub Enterprise Cloud environment acknowledge and sign off on the Hubber Handbook as part of the onboarding process. The Handbook includes the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics. | |
| | | 16 | GitHub management and peers evaluate and provide feedback to employees annually. | |

| CC1.0 | Common Criteria Related to Control Environment | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC1.2** | The board of directors demonstrates independence from management and exercises oversight of the development and performance of internal control. | 3 | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | No exceptions noted. |
| | | 4 | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | |
| **CC1.3** | Management establishes, with board oversight, structures, reporting lines, and appropriate authorities and responsibilities in the pursuit of objectives. | 3 | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | No exceptions noted. |
| | | 4 | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | |
| | | 5 | People Operations maintains a current organizational chart, outlining Hubbers' roles, which is made available to employees and contractors. Hubber responsibilities are communicated through documented job descriptions, stored in the respective teams' GitHub repository. | |
| | | 16 | GitHub management and peers evaluate and provide feedback to employees annually. | |

| CC1.0 | Common Criteria Related to Control Environment | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC1.4** | The entity demonstrates a commitment to attract, develop, and retain competent individuals in alignment with objectives. | 3 | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | No exceptions noted. |
| | | 6 | GitHub Security management evaluates the need for spending (headcount, tooling, or consultancy) to achieve security and compliance business objectives during GitHub's annual business planning and budgeting process. | |
| | | 7 | Interviewers evaluate employment candidates' abilities in the interview process against established job descriptions. The hiring manager then scores, tracks, and approves the fit to the role in the recruitment system. | |
| | | 8 | Employment candidates for GitHub roles undergo background screening as permissible by local laws and regulations. | |
| | | 16 | GitHub management and peers evaluate and provide feedback to employees annually. | |
| **CC1.5** | The entity holds individuals accountable for their internal control responsibilities in the pursuit of objectives. | 2 | The GitHub ISMS Policy 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 CSO reviews and approves the policy annually. | No exceptions noted. |
| | | 4 | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | |
| | | 16 | GitHub management and peers evaluate and provide feedback to employees annually. | |

## CC2.0 Common Criteria Related to Communication and Information

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC2.1** | The entity obtains or generates and uses relevant, quality information to support the functioning of internal control. | 3 | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | No exceptions noted. |
| | | 19 | The Security GRCC team assesses internal security controls annually through sample-based testing to help ensure control design and operation continue to meet defined criteria and system requirements. | |
| | | 21 | A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution. | |
| | | 22 | The Security Operations team scans internal and external systems monthly, reviews identified observations, and shares confirmed threats with responsible stakeholders. | |
| | | 24 | GitHub operates a bug bounty 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. | |
| | | 42 | 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. | |

| CC2.0 | Common Criteria Related to Communication and Information | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC2.2** | The entity internally communicates information, including objectives and responsibilities for internal control, necessary to support the functioning of internal control. | 2 | The GitHub ISMS Policy 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 CSO reviews and approves the policy annually. | No exceptions noted. |
| | | 4 | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | |
| | | 5 | People Operations maintains a current organizational chart, outlining Hubbers' roles, which is made available to employees and contractors. Hubber responsibilities are communicated through documented job descriptions, stored in the respective teams' GitHub repository. | |
| | | 10 | GitHub has published policies and procedures that outline employees' responsibilities for reporting security and operational failures, incidents, system problems, and complaints. | |
| | | 11 | GitHub personnel are required to complete security and privacy awareness training on an annual basis. | |
| | | 12 | The GitHub external website communicates to internal and external users the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints. | |

| CC2.0 | Common Criteria Related to Communication and Information | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC2.3** | The entity communicates with external parties regarding matters affecting the functioning of internal control. | 12 | The GitHub external website communicates to internal and external users the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints. | No exceptions noted. |
| | | 13 | Customer commitments and responsibilities are communicated through GitHub's Terms of Service or customer-provided contracts. | |
| | | 14 | GitHub posts notifications of new releases and customer-impacting changes to the GitHub blog. | |
| | | 41 | GitHub Support receives reports of security issues from customers via emails or through the web form. A ticketing system is used to document and track these issues to resolution. | |

## CC3.0 Common Criteria Related to Risk Assessment

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC3.1** | The entity specifies objectives with sufficient clarity to enable the identification and assessment of risks relating to objectives. | 1 | 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. | No exceptions noted. |
| | | 3 | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | |
| | | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | |
| **CC3.2** | The entity identifies risks to the achievement of its objectives across the entity and analyzes risks as a basis for determining how the risks should be managed. | 17 | GitHub tracks production components and third-party services to help ensure various aspects of the system inventory are reviewed and updated on a quarterly basis. | No exceptions noted. |
| | | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | |
| **CC3.3** | The entity considers the potential for fraud in assessing risks to the achievement of objectives. | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | No exceptions noted. |

## CC3.0 Common Criteria Related to Risk Assessment

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC3.4** | The entity identifies and assesses changes that could significantly impact the system of internal control. | 1 | 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. | No exceptions noted. |
| | | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | |
| | | 20 | Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repository. On an annual basis, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports. | |

| CC4.0 | Common Criteria Related to Monitoring Activities | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC4.1** | The entity selects, develops, and performs ongoing and/or separate evaluations to ascertain whether the components of internal control are present and functioning. | 19 | The Security GRCC team assesses internal security controls annually through sample-based testing to help ensure control design and operation continue to meet defined criteria and system requirements. | No exceptions noted. |
| | | 21 | A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution. | |
| | | 22 | The Security Operations team scans internal and external systems monthly, reviews identified observations, and shares confirmed threats with responsible stakeholders. | |
| **CC4.2** | The entity evaluates and communicates internal control deficiencies in a timely manner to those parties responsible for taking corrective action, including senior management and the board of directors, as appropriate. | 4 | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | No exceptions noted. |
| | | 23 | System owners remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process. | |

| CC5.0 | Common Criteria Related to Control Activities | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC5.1** | The entity selects and develops control activities that contribute to the mitigation of risks to the achievement of objectives to acceptable levels. | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | No exceptions noted. |
| | | 22 | The Security Operations team scans internal and external systems monthly, reviews identified observations, and shares confirmed threats with responsible stakeholders. | |
| | | 23 | System owners remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process. | |
| | | 24 | GitHub operates a bug bounty 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. | |
| **CC5.2** | The entity also selects and develops general control activities over technology to support the achievement of objectives. | 2 | The GitHub ISMS Policy 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 CSO reviews and approves the policy annually. | No exceptions noted. |
| | | 15 | GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes. | |

| CC5.0 | Common Criteria Related to Control Activities | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC5.3** | The entity deploys control activities through policies that establish what is expected and in procedures that put policies into action. | 2 | The GitHub ISMS Policy 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 CSO reviews and approves the policy annually. | No exceptions noted. |
| | | 15 | GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes. | |

| CC6.0 | Common Criteria Related to Logical and Physical Access Controls | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC6.1** | The entity implements logical access security software, infrastructure, and architectures over protected information assets to protect them from security events to meet the entity's objectives. | 17 | GitHub tracks production components and third-party services to help ensure various aspects of the system inventory are reviewed and updated on a quarterly basis. | 32: For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination. |
| | | 25 | A unique account and password are required to authenticate customers to their organization in Enterprise Cloud. | |
| | | 26 | Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token. | |
| | | 27 | Remote access to Enterprise Cloud production systems is restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which require 2FA. | |
| | | 28 | VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity. | |
| | | 31 | GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. 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. | |
| | | 32 | 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. | |
| | | 34 | Users are only able to access their personal repositories and the repositories where they are granted access. | |
| | | 36 | Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary. | |
| | | 52 | Customer Git repositories are encrypted at rest. | |
| | | 54 | Backups of customer Git repositories are performed on a real-time basis using an automated system. Backups of customer databases are encrypted during creation. | |

| CC6.0 | Common Criteria Related to Logical and Physical Access Controls | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC6.2** | Prior to issuing system credentials and granting system access, the entity registers and authorizes new internal and external users whose access is administered by the entity. For those users whose access is administered by the entity, user system credentials are removed when user access is no longer authorized. | 26 | Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token. | 32: For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination. |
| | | 29 | GitHub policy prohibits the use of shared administrative accounts for Enterprise Cloud production systems. Exceptions to the policy are documented in the exceptions repository. | |
| | | 31 | GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. 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. | |
| | | 32 | 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. | |
| | | 33 | Semiannually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm users with this access are restricted based on the principle of least privilege. | |

## CC6.0 Common Criteria Related to Logical and Physical Access Controls

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC6.3** | The entity authorizes, modifies, or removes access to data, software, functions, and other protected information assets based on roles, responsibilities, or the system design and changes, giving consideration to the concepts of least privilege and segregation of duties, to meet the entity's objectives. | 31 | GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. 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. | 32: For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination. |
| | | 32 | 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. | |
| | | 33 | Semiannually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm users with this access are restricted based on the principle of least privilege. | |
| | | 51 | 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. | |
| | | 58 | Server and database administrative privileges are restricted to the respective system owners. | |
| **CC6.4** | The entity restricts physical access to facilities and protected information assets (for example, data center facilities, backup media storage, and other sensitive locations) to authorized personnel to meet the entity's objectives. | 31 | GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. 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. | No exceptions noted. |
| | | 35 | The Infrastructure team reviews physical access to production data centers on an annual basis; any unapproved accounts are removed. | |
| **CC6.5** | The entity discontinues logical and physical protections over physical assets only after the ability to read or recover data and software from those assets has been diminished and is no longer required to meet the entity's objectives. | 30 | Data center media used to store production data are destroyed onsite. Certificates verifying media destruction are documented and retained. | No exceptions noted. |
| | | 39 | Full disk encryption is required on company laptops and is enforced through a centralized endpoint management system. | |

| CC6.0 | Common Criteria Related to Logical and Physical Access Controls | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC6.6** | The entity implements logical access security measures to protect against threats from sources outside its system boundaries. | 26 | Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token. | No exceptions noted. |
| | | 27 | Remote access to Enterprise Cloud production systems is restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which require 2FA. | |
| | | 28 | VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity. | |
| | | 36 | Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary. | |
| | | 37 | GitHub uses SSH to authenticate to back-end production resources through a bastion host. | |
| **CC6.7** | The entity restricts the transmission, movement, and removal of information to authorized internal and external users and processes and protects it during transmission, movement, or removal to meet the entity's objectives. | 37 | GitHub uses SSH to authenticate to back-end production resources through a bastion host. | No exceptions noted. |
| | | 38 | Production traffic is encrypted when transmitted over the public internet. | |
| | | 54 | Backups of customer Git repositories are performed on a real-time basis using an automated system. Backups of customer databases are encrypted during creation. | |
| | | 58 | Server and database administrative privileges are restricted to the respective system owners. | |

| CC6.0 | Common Criteria Related to Logical and Physical Access Controls | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC6.8** | The entity implements controls to prevent or detect and act upon the introduction of unauthorized or malicious software to meet the entity's objectives. | 40 | Threat detection software is installed on company endpoints to monitor for malicious software or unauthorized activity. | No exceptions noted. |
| | | 41 | GitHub Support receives reports of security issues from customers via emails or through the web form. A ticketing system is used to document and track these issues to resolution. | |
| | | 42 | 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. | |
| | | 56 | A configuration management tool maintains the state of systems to an approved baseline and reverts unapproved changes detected back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review. | |

| CC7.0 | Common Criteria Related to System Operations | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC7.1** | To meet its objectives, the entity uses detection and monitoring procedures to identify (1) changes to configurations that result in the introduction of new vulnerabilities, and (2) susceptibilities to newly discovered vulnerabilities. | 22 | The Security Operations team scans internal and external systems monthly, reviews identified observations, and shares confirmed threats with responsible stakeholders. | No exceptions noted. |
| | | 23 | System owners remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process. | |
| | | 24 | GitHub operates a bug bounty 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. | |
| | | 55 | GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures. | |
| | | 56 | A configuration management tool maintains the state of systems to an approved baseline and reverts unapproved changes detected back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review. | |
| **CC7.2** | The entity monitors system components and the operation of those components for anomalies that are indicative of malicious acts, natural disasters, and errors affecting the entity's ability to meet its objectives; anomalies are analyzed to determine whether they represent security events. | 21 | A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution. | No exceptions noted. |
| | | 42 | 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. | |

## CC7.0 Common Criteria Related to System Operations

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC7.3** | The entity evaluates security events to determine whether they could or have resulted in a failure of the entity to meet its objectives (security incidents) and, if so, takes actions to prevent or address such failures. | 41 | GitHub Support receives reports of security issues from customers via emails or through the web form. A ticketing system is used to document and track these issues to resolution. | No exceptions noted. |
| | | 43 | 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 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. | |
| | | 44 | 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. | |
| **CC7.4** | The entity responds to identified security incidents by executing a defined incident response program to understand, contain, remediate, and communicate security incidents, as appropriate. | 43 | 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 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. | No exceptions noted. |
| | | 44 | 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. | |

| CC7.0 | Common Criteria Related to System Operations | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC7.5** | The entity identifies, develops, and implements activities to recover from identified security incidents. | 43 | 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 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. | No exceptions noted. |

| CC8.0 | Common Criteria Related to Change Management | | | |
|---|---|---|---|---|
| | **Trust Services Criteria** | **Control #** | **Controls Specified by GitHub** | **Test Results** |
| **CC8.1** | The entity authorizes, designs, develops or acquires, configures, documents, tests, approves, and implements changes to infrastructure, data, software, and procedures to meet its objectives. | 45 | For new features and major changes, technical reviews are performed to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production. | 53: From February 25, 2022, to June 22, 2022,, real-time reviews of emergency deployments were not performed in a timely manner. |
| | | 46 | An independent reviewer evaluates and approves system change requests via a GitHub pull request. | |
| | | 47 | GitHub engineering code owners create and execute automated regression test cases to address functionality and security requirements. | |
| | | 48 | Application changes pass automated continuous integration testing prior to deployment to the production environment. | |
| | | 49 | 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. | |
| | | 50 | Test environments are logically separated from production environments. | |
| | | 51 | 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. | |
| | | 53 | The Product Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems. | |
| | | 55 | GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures. | |

| CC8.0 | Common Criteria Related to Change Management | | | |
|---|---|---|---|---|
| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
| | | 56 | A configuration management tool maintains the state of systems to an approved baseline and reverts unapproved changes detected back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review. | |
| | | 57 | Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification. | |

## CC9.0 Common Criteria Related to Risk Mitigation

| | Trust Services Criteria | Control # | Controls Specified by GitHub | Test Results |
|---|---|---|---|---|
| **CC9.1** | The entity identifies, selects, and develops risk mitigation activities for risks arising from business disruption. | 1 | 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. | No exceptions noted. |
| | | 18 | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | |
| **CC9.2** | The entity assesses and manages risks associated with vendors and business partners. | 17 | GitHub tracks production components and third-party services to help ensure various aspects of the system inventory are reviewed and updated on a quarterly basis. | No exceptions noted. |
| | | 20 | Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repository. On an annual basis, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports. | |

## B. Description of Test of Controls and Results

Test procedures performed in connection with determining the operating effectiveness of controls detailed here in Section IV are described below:

| Test Procedure | Description |
|---|---|
| **Inquiry** | Inquiry of appropriate personnel and corroboration with management. |
| **Observation** | Observation of the application, performance, or existence of the control. |
| **Inspection** | Inspection of documents and reports indicating performance of the control. |
| **Reperformance** | Reperformance of the control. |

In addition, we evaluated whether the information was sufficiently reliable for our purposes by obtaining evidence about the accuracy and completeness of such information and evaluating whether the information was sufficiently precise and detailed for our purposes.

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **1** | 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. | Inquired of the Director, Security GRCC to confirm GitHub had established a formal Security Risk Management Policy to identify, assess, and manage security and security compliance-related risks; and GitHub Security leadership reviewed and approved the Security Risk Management Policy annually. | No exceptions noted. |
| | | Inspected the Security Risk Management Policy to determine whether this policy defined the risk management program; and whether it included steps for assessing and managing security and compliance risks. | No exceptions noted. |
| | | Inspected the Security Risk Management Policy revision log to determine whether Security leadership reviewed and approved this document during the examination period. | No exceptions noted. |
| **2** | The GitHub ISMS Policy 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 CSO reviews and approves the policy annually. | Inquired of the Director, Security GRCC to confirm the ISMS Policy defined information security practices including information system management, access controls, software development, and the definition of roles and responsibilities; the policy was published to internal personnel via the GitHub intranet; and the CSO reviewed and approved the policy annually. | No exceptions noted. |
| | | Inspected the ISMS Policy to determine whether this policy defined information security practices and roles and responsibilities; whether this policy was published to internal personnel via the GitHub intranet; and whether the CSO reviewed and approved this policy during the examination period. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **3** | GitHub senior leadership sets company-wide objectives and key results semiannually. Operational objectives and key results defined across the domains in scope tie back to meeting those company-wide organizational objectives, including security and risk objectives. Senior leadership reports and reviews progress towards meeting those objectives semiannually. | Inquired of the Director, Security GRCC to confirm GitHub leadership set company-wide objectives and key results semiannually and operational objectives and key results tied back to meet those company-wide organizational objectives, including security and risk objectives; and semiannually senior leadership reported and reviewed progress towards meeting those objectives. | No exceptions noted. |
| | | Inspected the company-wide objectives and key results for a sample semiannual period to determine whether these objectives and results were defined and included security and risk topics for this period. | No exceptions noted. |
| | | Inspected a sample semiannual report from the Security GRCC team to determine whether senior leadership was informed of progress toward their respective objectives and key results during this semiannual period. | No exceptions noted. |
| **4** | The Director, Security GRCC reports the status of control operating effectiveness and risks to senior leadership quarterly. Senior leadership is independent from control operations. | Inquired of the Director, Security GRCC to confirm status reports including control operating effectiveness and risks were communicated to senior leadership quarterly; and senior leadership was independent from control operations. | No exceptions noted. |
| | | Inspected status reports sent to senior leadership for a sample of quarters to determine whether these reports included status on control operating effectiveness and risks. | No exceptions noted. |
| | | Inspected the company organizational chart and job descriptions to determine whether senior leadership was independent from the operation of controls. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **5** | People Operations maintains a current organizational chart, outlining Hubbers' roles, which is made available to employees and contractors. Hubber responsibilities are communicated through documented job descriptions, stored in the respective teams' GitHub repository. | Inquired of the Manager, Security GRCC to confirm People Operations maintained an organizational chart outlining Hubbers' roles; the organizational chart was made available to employees and contractors; and Hubber job responsibilities were communicated through documented job descriptions stored in the respective teams' GitHub repository. | No exceptions noted. |
| | | Inspected the organizational chart revision history to determine whether GitHub maintained an organizational chart outlining Hubbers' roles. | No exceptions noted. |
| | | Inspected the HR system portal to determine whether the organizational chart was made available to employees and contractors. | No exceptions noted. |
| | | Inspected a sample of team repositories to determine whether associated job descriptions were documented and job responsibilities were defined. | No exceptions noted. |
| **6** | GitHub Security management evaluates the need for spending (headcount, tooling, or consultancy) to achieve security and compliance business objectives during GitHub's annual business planning and budgeting process. | Inquired of the Director, Security GRCC to confirm GitHub Security management evaluated the need for spending (headcount, tooling, or consultancy) to achieve security and compliance business objectives during GitHub's annual business planning and budgeting process. | No exceptions noted. |
| | | Inspected the security budget to determine whether GitHub Security management evaluated the need for spending (headcount, tooling, or consultancy) to achieve security and compliance business objectives during GitHub's annual business planning and budgeting process. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| 7 | Interviewers evaluate employment candidates’ abilities in the interview process against established job descriptions. The hiring manager then scores, tracks, and approves the fit to the role in the recruitment system. | Inquired of the Manager, Talent Acquisition to confirm interviewers evaluated employment candidates' abilities in the interview process against established job descriptions; and the hiring manager scored, tracked, and approved the fit to the role in the recruitment system. | No exceptions noted. |
| | | Inspected job descriptions and candidate evaluations for a sample of new hires to determine whether the hiring manager scored, tracked, and approved these candidates in the recruitment system based on interviews performed. | No exceptions noted. |
| 8 | Employment candidates for GitHub roles undergo background screening as permissible by local laws and regulations. | Inquired of the Manager, Talent Acquisition to confirm employment candidates for GitHub roles underwent background screening as permissible by local laws and regulations. | No exceptions noted. |
| | | Inspected the background screening results for a sample of new employees to determine whether these employees underwent background screening as part of the onboarding process. | No exceptions noted. |
| 9 | Individuals with access to the GitHub Enterprise Cloud environment acknowledge and sign off on the Hubber Handbook as part of the onboarding process. The Handbook includes the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics. | Inquired of the Manager, Talent Acquisition to confirm Hubbers acknowledged and signed-off on the Hubber Handbook as part of the onboarding process; and the Handbook included the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics. | No exceptions noted. |
| | | Inspected the Hubber Handbook to determine whether it included the Standards of Conduct, Security Policy Awareness and Responsibilities, and other information security topics. | No exceptions noted. |
| | | Inspected the onboarding documents for a sample of new employees to determine whether these employees signed the Hubber Handbook as part of the onboarding process. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **10** | GitHub has published policies and procedures that outline employees' responsibilities for reporting security and operational failures, incidents, system problems, and complaints. | Inquired of the Director, Security GRCC to confirm GitHub had published policies and procedures that outlined employees' responsibilities for reporting security and operational failures, incidents, system problems, and complaints. | No exceptions noted. |
| | | Inspected the Employee Handbook and Security Risk Reporting SOP to determine whether GitHub had published policies and procedures that outlined employees' responsibilities to report security and operational failures, incidents, system problems, and complaints. | No exceptions noted. |
| | | Inspected the GitHub intranet to determine whether the Employee Handbook and Security Risk Reporting SOP were published internally. | No exceptions noted. |
| **11** | GitHub personnel are required to complete security and privacy awareness training on an annual basis. | Inquired of the Senior GRCC Analyst to confirm GitHub personnel were required to complete security and privacy training annually. | No exceptions noted. |
| | | Inspected training records in the learning management system for a sample of GitHub personnel to determine whether these individuals completed training during the examination period. | No exceptions noted. |
| **12** | The GitHub external website communicates to internal and external users the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints. | Inquired of the Director, Security GRCC to confirm the GitHub external website communicated to internal and external users the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints. | No exceptions noted. |
| | | Inspected github.com content to determine whether it communicated to internal and external users the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| 13 | Customer commitments and responsibilities are communicated through GitHub’s Terms of Service or customer-provided contracts. | Inquired of the Senior Director, Sales Operations to confirm customer commitments and responsibilities were communicated through GitHub's Terms of Service or customer-provided contracts. | No exceptions noted. |
| | | Inspected signed Terms of Service or customer-provided contracts for a sample of customers to determine whether customer commitments and responsibilities were communicated through these documents. | No exceptions noted. |
| 14 | GitHub posts notifications of new releases and customer-impacting changes to the GitHub blog. | Inquired of the Technical Project Manager to confirm GitHub posted notifications of new releases and customer-impacting changes via the GitHub blog. | No exceptions noted. |
| | | Inspected blog.github.com to determine whether GitHub posted notifications of new releases and customer-impacting changes via the GitHub blog. | No exceptions noted. |
| | | Inspected blog posts for a sample of code changes to determine whether GitHub posted notifications of these releases and customer-impacting changes to the GitHub blog. | No exceptions noted. |
| 15 | GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes. | Inquired of the Director, Security GRCC to confirm GitHub maintained and communicated key security standards and procedures on the GitHub intranet; and these standards and procedures addressed the security of systems, facilities, data, personnel, and processes. | No exceptions noted. |
| | | Inspected the GitHub intranet permissions to determine whether key security standards and procedures were made available and communicated to employees. | No exceptions noted. |
| | | Inspected key security standards and procedures on the GitHub intranet to determine whether GitHub maintained these security standards and procedures that addressed the security of systems, facilities, data, personnel, and processes. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **16** | GitHub management and peers evaluate and provide feedback to employees annually. | Inquired of the Director, Security GRCC to confirm GitHub management and peers evaluated and provided feedback to employees annually. | No exceptions noted. |
| | | Inspected management and peer review reports for a sample of employees to determine whether GitHub management and peers evaluated and provided feedback to these employees during the examination period. | No exceptions noted. |
| **17** | GitHub tracks production components and third-party services to help ensure various aspects of the system inventory are reviewed and updated on a quarterly basis. | Inquired of the Director, Security GRCC to confirm GitHub tracked production components and third-party services to help ensure various aspects of the system inventory were reviewed and updated on a quarterly basis. | No exceptions noted. |
| | | Inspected the system inventory review documentation for a sample of quarters to determine whether GitHub tracked production components and third-party services to verify various aspects of the system inventory were tracked, reviewed, and updated for these quarters. | No exceptions noted. |
| **18** | On an annual basis, GitHub performs a risk assessment that identifies, tracks, and monitors changes to Enterprise Cloud security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. | Inquired of the Director, Security GRCC to confirm GitHub performed an annual risk assessment that identified, tracked, and monitored changes to Enterprise Cloud security, related compliance risk factors, and fraud; and risks identified were documented in a repository where the risks and mitigation actions were tracked to resolution. | No exceptions noted. |
| | | Inspected the results from the most recent risk assessment documented in the risk reporting repository to determine whether GitHub performed a risk assessment during the examination period; and whether it identified, tracked, and monitored changes to Enterprise Cloud security, related compliance risk factors, and fraud. | No exceptions noted. |
| | | Inspected the risk reporting repository and associated risk documentation for a sample of identified risks to determine whether these risks were tracked and mitigated. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| 19 | The Security GRCC team assesses internal security controls annually through sample-based testing to help ensure control design and operation continue to meet defined criteria and system requirements. | Inquired of the Director, Security GRCC to confirm the Security GRCC team assessed internal security controls annually through sample-based testing to help ensure control design and operation continued to meet defined criteria and system requirements. | No exceptions noted. |
| | | Inspected internal control test results to determine whether the Security GRCC team assessed internal security controls during the examination period through sample-based testing to help ensure control design and operation continued to meet defined criteria and system requirements. | No exceptions noted. |
| 20 | Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repository. On an annual basis, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports. | Inquired of the Director, Security GRCC to confirm GitHub reviewed vendor security compliance prior to engagement a new vendor and documented the results in the vendor security repository; and GitHub assessed the controls implemented at subservice organizations that received or stored customer data on an annual basis through the review of relevant subservice organization attestation reports. | No exceptions noted. |
| | | Inspected the vendor security compliance review documentation for a sample of new vendors to determine whether GitHub reviewed vendor security compliance for these vendors prior to engagement. | No exceptions noted. |
| | | Inspected the control assessments for a sample of subservice providers that received or stored customer data to determine whether GitHub assessed subservice organization controls during the examination period through the review of these subservice organizations' attestation reports. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **21** | A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution. | Inquired of the Director of Application Security Engineering to confirm a third party performed an annual application penetration assessment; and Security Operations triaged and tracked identified issues through to resolution. | No exceptions noted. |
| | | Inspected the penetration test report to determine whether a third party performed an application penetration assessment during the examination period. | No exceptions noted. |
| | | Inspected the penetration test report and tracking documentation to determine whether Security Operations triaged and tracked identified issues through to resolution. | No exceptions noted. |
| **22** | The Security Operations team scans internal and external systems monthly, reviews identified observations, and shares confirmed threats with responsible stakeholders. | Inquired of the Manager, Security Operations to confirm Security Operations scanned internal and external systems monthly, reviewed identified observations, and shared confirmed threats with responsible stakeholders. | No exceptions noted. |
| | | Inspected the vulnerability scanning configuration to determine whether internal and external security vulnerability scanning was configured to execute on a monthly basis. | No exceptions noted. |
| | | Inspected ticketing system documentation for a sample of identified vulnerabilities to determine whether Security Operations reviewed these observations and shared confirmed threats with responsible stakeholders. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **23** | System owners remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process. | Inquired of the Manager, Security Operations to confirm system owners remediated vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities; and vulnerabilities that could not be remediated within the required SLA were handled via the documented exception process. | No exceptions noted. |
| | | Inspected the Vulnerability Management Process Procedure to determine whether system owners were required to remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities; and whether vulnerabilities that could not be remediated within the required SLA were handled via the documented exception process. | No exceptions noted. |
| | | Inspected tickets for a sample of vulnerabilities to determine whether these vulnerabilities were remediated within SLA or were handled via the documented exception process; and whether these remediated vulnerabilities were completed in accordance with established SLAs based on the severity of the vulnerabilities. | No exceptions noted. |
| **24** | GitHub operates a bug bounty 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. | Inquired of the Security Engineering Senior Manager to confirm GitHub operated a bug bounty program; on-call security resources monitored the submissions and triaged accordingly; and for legitimate bugs, Security informed the relevant engineers and the bug was tracked in GitHub issues to resolution. | No exceptions noted. |
| | | Observed the functionality of the bug bounty platform to determine whether researchers were able to report bugs. | No exceptions noted. |
| | | Inspected tracking tickets for a sample of bugs in the bug tracking repository to determine whether these bugs were triaged and tracked to resolution. | No exceptions noted. |
| **25** | A unique account and password are required to authenticate customers to their organization in Enterprise Cloud. | Inquired of the Engineering Manager to confirm a unique account and password were required to authenticate customers to their organization in Enterprise Cloud. | No exceptions noted. |
| | | Inspected the authentication code configuration to determine whether a unique account and password were required to authenticate customers to their organization in Enterprise Cloud. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **26** | Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token. | Inquired of the Security Operations Manager to confirm remote access to internal administration tools was restricted through unique account IDs, passwords, and a one-time token. | No exceptions noted. |
| | | Observed the Security Operations Manager authenticate to a sample of the internal administration tools to determine whether these administration tools were restricted through unique account IDs, passwords, and a one-time token. | No exceptions noted. |
| | | Inspected the production user access listing, the VPN authentication configuration, and the bastion host 2FA configuration to determine whether remote access to internal administration tools required unique account IDs, passwords, and a one-time token. | No exceptions noted. |
| **27** | Remote access to Enterprise Cloud production systems is restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which require 2FA. | Inquired of the Security Operations Manager to confirm remote access to Enterprise Cloud production systems was restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which required 2FA. | No exceptions noted. |
| | | Inspected the authentication configuration to determine whether remote access to Enterprise Cloud production systems was restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which required 2FA. | No exceptions noted. |
| | | Observed the Security Operations Manager authenticate to Enterprise Cloud production systems to determine whether either a VPN, bastion host, or SAML connection was required. | No exceptions noted. |
| | | Observed the Security Operations Manager connect over the VPN, the bastion host, and SAML to determine whether the VPN, bastion host, and SAML required 2FA. | No exceptions noted. |
| **28** | VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity. | Inquired of the Security Operations Manager to confirm VPN and bastion hosts were configured to require user connections to reauthenticate after 24 hours of inactivity. | No exceptions noted. |
| | | Inspected the VPN and bastion hosts inactivity configuration to determine whether VPN and bastion host user connections were required to reauthenticate after 24 hours of inactivity. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **29** | GitHub policy prohibits the use of shared administrative accounts for Enterprise Cloud production systems. Exceptions to the policy are documented in the exceptions repository. | Inquired of the Security Operations Manager to confirm GitHub policy prohibited the use of shared administrative accounts for Enterprise Cloud production systems; and exceptions to this policy were documented in the exceptions repository. | No exceptions noted. |
| | | Inspected the Hubber Handbook to determine whether GitHub prohibited the use of shared administrative accounts for Enterprise Cloud production systems. | No exceptions noted. |
| | | Inspected the user access listing to Enterprise Cloud production systems to determine whether GitHub identified shared administrative accounts for Enterprise Cloud production systems. | No exceptions noted. |
| | | Inspected account user lists and the exceptions repository to determine whether shared account exceptions were documented. | No exceptions noted. |
| **30** | Data center media used to store production data are destroyed onsite. Certificates verifying media destruction are documented and retained. | Inquired of the Site Reliability Engineer to confirm data center media used to store production data were destroyed onsite; and certificates were documented and retained providing verification of media destruction. | No exceptions noted. |
| | | Inspected certificates of destruction for a sample of discontinued media storing production data to determine whether these devices were destroyed, and whether the certificate verified the destruction. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **31** | GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. 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. | Inquired of the Security Operations Manager to confirm GitHub's internal systems were configured to automatically provision logical and physical access based on the user's job role; and GitHub's internal systems were configured to automatically provision non role-based entitlements only after the user's manager reviewed and approved the access request. | No exceptions noted. |
| | | Inspected the system configurations for the role-based entitlements to determine whether GitHub's internal systems were configured to automatically provision logical and physical access based on the user's job role. | No exceptions noted. |
| | | Inspected the system configurations for the non-role-based entitlements to determine whether GitHub's internal systems were configured to automatically provision logical and physical access only after the user's manager reviewed and approved the access request. | No exceptions noted. |
| | | Inspected a pull request for a sample user who received role-based entitlements to determine whether these user entitlements were granted to this user based on the user's role. | No exceptions noted. |
| | | Inspected a pull request for a sample user who received non role-based entitlements to determine whether these user entitlements were granted only after the user's manager reviewed and approved the access request. | No exceptions noted. |
| | | Inspected the entitlements repository revision history and the change population to determine whether these configurations were in place throughout the examination period. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **32** | 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. | Inquired of the Security Operations Manager to confirm the Security Operations team revoked logical production access and Production Engineering revoked physical production access for terminated personnel (employee and contractors) within 24 hours of termination. | No exceptions noted. |
| | | Inspected termination checklists for a sample of terminated personnel to determine whether Security Operations documented the revocation of production access and Production Engineering documented the revocation of physical access for these personnel within 24 hours of their termination. | For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination.<br><br>See *Section V - Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report* for management response to the noted exception. |
| | | Inspected the production user access levels and physical access levels for a sample of terminated users to determine whether their physical and logical access to production systems was revoked. | No exceptions noted. |
| **33** | Semiannually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm users with this access are restricted based on the principle of least privilege. | Inquired of the Director, Security GRCC to confirm the Security Operations team reviewed elevated access permissions semiannually to confirm users with this access were restricted based on the principle of least privilege; and the access reviewed was reauthorized at the end of the review. | No exceptions noted. |
| | | Inspected review documentation from both semiannual security reviews performed during the examination period to determine whether the Security Operations team reviewed and reauthorized elevated access permissions. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **34** | Users are only able to access their personal repositories and the repositories where they are granted access. | Inquired of the Engineering Manager to confirm users were only able to access their personal repositories and the repositories where they were granted access. | No exceptions noted. |
| | | Inspected the GitHub.com repository configuration to determine whether user access was limited to a user's personal repositories and repositories where they were granted access. | No exceptions noted. |
| | | Observed the repository access restrictions in the system to determine whether a test user was properly restricted from accessing a test repository where the user did not have access. | No exceptions noted. |
| **35** | The Infrastructure team reviews physical access to production data centers on an annual basis; any unapproved accounts are removed. | Inquired of the Director, Security GRCC to confirm the Infrastructure team reviewed physical access to production data centers on an annual basis; and any unapproved accounts were removed. | No exceptions noted. |
| | | Inspected physical access review documentation to determine whether the Infrastructure team reviewed physical access to production data centers during the examination period. | No exceptions noted. |
| | | Inspected physical access lists and physical access review documentation to determine whether unapproved accounts were removed. | No exceptions noted. |
| **36** | Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary. | Inquired of the Site Reliability Engineer to confirm network firewalls restricted external points of connectivity and prevented unauthorized traffic from beyond the system boundary. | No exceptions noted. |
| | | Inspected firewall configurations to determine whether they restricted external points of connectivity, and whether they were configured to prevent unauthorized traffic from beyond the system boundary. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **37** | GitHub uses SSH to authenticate to back-end production resources through a bastion host. | Inquired of the Site Reliability Engineer to confirm GitHub used SSH to authenticate to back-end production resources through a bastion host. | No exceptions noted. |
| | | Inspected the authentication configuration to determine whether GitHub uses SSH to authenticate to back-end production resources through a bastion host. | No exceptions noted. |
| | | Observed a user authenticate to back-end production resources to determine whether SSH was required for the user to authenticate. | No exceptions noted. |
| **38** | Production traffic is encrypted when transmitted over the public internet. | Inquired of the Security Operations Manager to confirm production traffic was encrypted when transmitted over the public internet. | No exceptions noted. |
| | | Inspected www.github.com and the corresponding encryption certificate to determine whether production traffic over the internet was encrypted. | No exceptions noted. |
| **39** | Full disk encryption is required on company laptops and is enforced through a centralized endpoint management system. | Inquired of the IT Support Manager to confirm full disk encryption was required on company laptops and was enforced through a centralized endpoint management system. | No exceptions noted. |
| | | Inspected the central endpoint management system configuration to determine whether full disk encryption on company laptops was enforced. | No exceptions noted. |
| **40** | Threat detection software is installed on company endpoints to monitor for malicious software or unauthorized activity. | Inquired of the IT Support Manager to confirm threat detection software was installed on company endpoints to monitor for malicious software or unauthorized activity. | No exceptions noted. |
| | | Inspected the threat detection software configuration to determine whether threat detection software was installed on company endpoints to monitor for malicious or unauthorized activity. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **41** | GitHub Support receives reports of security issues from customers via emails or through the web form. A ticketing system is used to document and track these issues to resolution. | Inquired of the SIRT Manager to confirm GitHub Support received reports of security issues from customers via emails or through the web form; and a ticketing system was used to document and track these issues to resolution. | No exceptions noted. |
| | | Inspected the customer support communication channels to determine whether security events from customer emails or web requests were available and recorded. | No exceptions noted. |
| | | Inspected tickets for a sample of issues to determine whether these issues were documented and tracked to resolution. | No exceptions noted. |
| **42** | 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. | Inquired of the SIRT Manager to confirm logging and monitoring tools were used to collect telemetry and event logs from network devices, applications, and systems; and security alerts were generated and communicated to designated responders, and prioritized in a specific GitHub Security team repository for triage, tracking, and remediation. | No exceptions noted. |
| | | Inspected the logging and monitoring tool configuration to determine whether this tool was configured to capture telemetry and event logs from network devices, applications, and systems. | No exceptions noted. |
| | | Inspected the security monitoring tools and security event repositories for a sample of alerts to determine whether these security alerts were prioritized and communicated to designated responders for follow up. | No exceptions noted. |
| | | Inspected the GitHub Security team notification channel and the security event repository for a sample of alerts to determine whether these events were logged and tracked to remediation. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **43** | 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 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. | Inquired of the SIRT Manager to confirm an Incident Response Plan was documented and used when an incident was identified or reported; escalation procedures were included to address breach or privacy considerations depending on the severity of the incident; roles were assigned in the Incident Response Plan, and an incident response process was followed until the incident was remediated and documented; root cause was assessed and addressed as part of the incident response process; and incident post-mortems were documented and included lessons learned. | No exceptions noted. |
| | | Inspected the GitHub Security Incident Response Plan to determine whether the plan included steps to address breach or privacy considerations, identify incident response roles, assess root cause, and outline the process to track an incident to remediation. | No exceptions noted. |
| | | Inspected the incident response reports for a sample of incidents to determine whether the documentation for these incidents included the root cause, remediation steps, resolution, and a post-mortem, including lessons learned. | No exceptions noted. |
| **44** | 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. | Inquired of the SIRT Manager to confirm security personnel were notified 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. |
| | | Inspected the SIRT team communication channels to determine whether they were notified of security events through system monitoring alerts and reports from humans. | No exceptions noted. |
| | | Inspected the SIRT team communication channels for a sample of security events to determine whether these security events were surfaced to responders based on severity as defined in formally documented procedures. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **45** | For new features and major changes, technical reviews are performed to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production. | Inquired of the Director, Security GRCC to confirm technical reviews were performed to assess security, data, and architecture risk for new features and major changes; and cross functional approval was required prior to release of changes to production. | No exceptions noted. |
| | | Inspected the Software Development Lifecycle Policy to determine whether this policy required technical reviews assessing security, data, and architecture risk for new features and major changes, and whether cross functional approval was required prior to release of changes to production. | No exceptions noted. |
| | | Inspected application security reviews for a sample of new features and major changes to determine whether technical reviews were performed to assess security, data, and architecture risk for these changes, and whether cross functional approval was provided prior to release of these changes to production. | No exceptions noted. |
| **46** | An independent reviewer evaluates and approves system change requests via a GitHub pull request. | Inquired of the Director, Software Engineering to confirm an independent reviewer evaluated and approved system change requests via a GitHub pull request. | No exceptions noted. |
| | | Observed and inspected the pull request documentation for a sample of code changes to determine whether an independent reviewer evaluated and approved these changes. | No exceptions noted. |
| **47** | GitHub engineering code owners create and execute automated regression test cases to address functionality and security requirements. | Inquired of the Technical Project Manager to confirm GitHub engineering code owners created and executed automated regression test cases to address functionality and security requirements. | No exceptions noted. |
| | | Inspected the automated regression test cases for a sample of code changes to determine whether GitHub engineering code owners created and executed test cases to address functionality and security requirements for these changes. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **48** | Application changes pass automated continuous integration testing prior to deployment to the production environment. | Inquired of the Technical Project Manager to confirm application changes passed automated continuous integration testing prior to deployment to the production environment. | No exceptions noted. |
| | | Inspected pull request documentation for a sample of changes to determine whether these application changes passed automated continuous integration testing prior to deployment to the production environment. | No exceptions noted. |
| **49** | 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. | Inquired of the Director, Application Security Engineering to confirm a security code analysis tool scanned code during commits and merges, the scanning tool posted potential vulnerabilities on the change pull request, the developer assessed the potential vulnerabilities, and a peer reviewer followed up on the items during change review and approval. | No exceptions noted. |
| | | Inspected automated code security analysis results for a sample of code changes to determine whether an automated code security analysis was performed for these changes during commits and merges. | No exceptions noted. |
| | | Inspected pull requests for a sample of code changes to determine whether potential vulnerabilities identified for these changes were posted on the change pull request; whether the developer assessed the potential vulnerabilities; and whether a peer reviewer followed up on these items during change review and approval. | No exceptions noted. |
| **50** | Test environments are logically separated from production environments. | Inquired of the Technical Project Manager to confirm test environments were logically separated from production environments. | No exceptions noted. |
| | | Inspected the configuration of production and test environment clusters to determine whether these environments were logically separated. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **51** | 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. | Inquired of the Technical Project Manager to confirm 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. |
| | | Inspected branch protection system configurations for the GitHub and puppet repositories to determine whether system configurations enforced peer reviews and integration tests for these repositories prior to the deployment of changes to the production environment and alerted code owners of any forced deployments. | No exceptions noted. |
| | | Inspected tickets for a sample of code deployments to the production environment to determine whether peer reviews and integration tests were completed prior to the deployment of these changes to the production environment. | No exceptions noted. |
| | | Inspected an alert for a sample forced deployment to determine whether the assigned code owner was alerted of the forced deployment. | No exceptions noted. |
| **52** | Customer Git repositories are encrypted at rest. | Inquired of the Security Operations Manager to confirm customer Git repositories were encrypted at rest. | No exceptions noted. |
| | | Inspected customer Git repository encryption configurations for a sample of customers to determine whether customer Git repositories were encrypted. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **53** | The Product Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems. | Inquired of the Technical Project Manager to confirm emergency deployments were monitored on a real-time basis, and post-mortem reviews were performed for emergency changes that impacted production systems. | No exceptions noted. |
| | | Inspected the Software Development Lifecycle Policy to determine whether this policy defined requirements for real-time emergency deploy monitoring, and post-mortem reviews for emergency changes impacting production systems. | No exceptions noted. |
| | | Inspected deploy monitoring tools to determine whether the Product Engineering team monitored emergency deployments on a real-time basis. | No exceptions noted. |
| | | Inspected the post-mortem reviews for a sample of emergency code changes to determine whether a post-mortem review was performed for these changes. | From February 25, 2022, to June 22, 2022, real-time reviews of emergency deployments were not performed in a timely manner.<br><br>See *Section V - Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report* for management response to the noted exception. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **54** | Backups of customer Git repositories are performed on a real-time basis using an automated system. Backups of customer databases are encrypted during creation. | Inquired of the Senior Software Engineer to confirm customer Git repositories were backed up in real time using an automated system; and backups of customer databases were encrypted during creation. | No exceptions noted. |
| | | Inspected the backup configuration to determine whether customer Git repositories were configured to be backed up on a real-time basis using an automated system. | No exceptions noted. |
| | | Inspected the backup encryption configuration to determine whether customer database backups were configured to be automatically encrypted during the backup creation process. | No exceptions noted. |
| | | Inspected backup storage system logs to determine whether backups were created throughout the period. | No exceptions noted. |
| **55** | GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures. | Inquired of the Director, Security GRCC to confirm GitHub had defined hardening standards based on GitHub security hardening practices; and the Security Operations team reviewed and approved changes to the standard hardening procedures. | No exceptions noted. |
| | | Inspected standard hardening procedures to determine whether GitHub had defined hardening standards based on its security hardening practices. | No exceptions noted. |
| | | Inspected revision history approval documentation for a sample of changes to standard hardening procedures to determine whether the Security Operations team reviewed and approved these changes to the standard hardening procedures. | No exceptions noted. |

| Control # | Controls Specified by GitHub | Tests Performed by Moss Adams LLP | Test Results |
|---|---|---|---|
| **56** | A configuration management tool maintains the state of systems to an approved baseline and reverts unapproved changes detected back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review. | Inquired of the Site Reliability Engineer and Engineering Manager to confirm a configuration management tool maintained the state of systems to an approved state and reverted unapproved changes detected back to the required configuration; and changes to the approved configuration baseline were completed through pull requests requiring peer review. | No exceptions noted. |
| | | Observed an unapproved test change made to a configuration file to determine whether the configuration management tool would revert the configuration back to the approved baseline. | No exceptions noted. |
| | | Inspected change log information for a sample of changes to the approved configurations to determine whether these changes were peer reviewed through pull requests. | No exceptions noted. |
| **57** | Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification. | Inquired of the Director, Security GRCC to confirm code commits, whether direct commits or through a pull request, were hashed; and the hash was displayed with the commit to facilitate code integrity verification. | No exceptions noted. |
| | | Observed the submission of a test direct code commit and a test pull request to determine whether these test commits were hashed. | No exceptions noted. |
| | | Inspected test commits for a sample of changes to determine whether the hash was visible with these commits. | No exceptions noted. |
| **58** | Server and database administrative privileges are restricted to the respective system owners. | Inquired of the Data Infrastructure Engineer to confirm server and database administrative privileges are restricted to the respective system owners. | No exceptions noted. |
| | | Inspected access listings for servers and databases and employee records to determine whether server and database administrative privileges were restricted to the respective system owners. | No exceptions noted. |

## V. Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report

The following information is provided for informational purposes only and has not been subjected to the procedures applied in the examination. Accordingly, Moss Adams expresses no opinion on the following information.

### A. Management's Response to Identified Testing Exceptions

| Control # | Controls Specified by GitHub | Exception Noted by Moss Adams LLP | GitHub Management Response |
|---|---|---|---|
| **32** | 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. | For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination. | GitHub's employee termination process is comprised of two main sub-processes:<br><br>(1) HR's process to identify an employee/contractor's date of termination and configure that date manually within GitHub's HRIS;<br>(2) Security Operations' automated process to receive the notice of termination (systematically identified upon input of the termination date to the HRIS) and automatically (via system configuration and automation) remove access to GitHub's internal systems.<br><br>With regard to the control deviation noted by the Service Auditor, the manual sub-process discussed above was delayed due to miscommunication regarding the effective termination date of the employee in question. As soon as IT was able to confirm the details of the termination, the correct termination date was input/updated within the HRIS and the automated sub-process executed without delay or exception. Additionally, upon further inspection of the user's internal system access and account activity, GitHub Management determined no inappropriate actions were taken by the employee during the period as a result of the deviation noted by the Service Auditor. |

| Control # | Controls Specified by GitHub | Exception Noted by Moss Adams LLP | GitHub Management Response |
|---|---|---|---|
| **53** | The Product Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems. | From February 25, 2022, to June 22, 2022, real-time reviews of emergency deployments were not performed in a timely manner. | With regard to the deviation noted by the Service Auditor, GitHub Management performed a review of all emergency deployments performed during the examination period and determined that post-mortem reviews were performed for all emergency deployments. GitHub Management also determined there was a temporary pause in automated notifications triggering the deployment reviews; the automated alerting functionality was remediated during the examination period and is functioning appropriately. Notwithstanding the deviation noted by the Service Auditor, all other preventive change management controls tested by the Service Auditor for the current examination period were designed and operating effectively throughout the period. |

MOSSADAMS