Proprietary & Confidential

GitHub

Enterprise Cloud

SOC 3
Relevant to Security

Integrated SOC 3 Report Prepared in Accordance with the AICPA Attestation Standards and IAASB ISAE No. 3000 (Revised) Standards

OCTOBER 1, 2022 TO SEPTEMBER 30, 2023

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 assertion in Section II titled "GitHub's Assertion" (assertion) that the controls within GitHub's Enterprise Cloud (system) were effective throughout the period October 1, 2022 to September 30, 2023, to provide reasonable assurance that GitHub's 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 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 organization. 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. 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 also provided the accompanying assertion about the effectiveness of controls within the system. When preparing its assertion, GitHub is responsible for selecting, and identifying in its assertion, the applicable trust services criteria and for having a reasonable basis for its assertion by performing an assessment of the effectiveness of the controls within the system.

Service Auditor's Responsibilities

Our responsibility is to express an opinion, based on our examination, on whether management's assertion that controls within the system were effective throughout the period to provide reasonable assurance that the service organization's service commitments and system requirements were achieved based on the applicable trust services criteria. 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 3000 (Revised), *Assurance Engagements Other Than Audits or Reviews of Historical Financial Information*, issued by the International Auditing and Assurance Standards Board. Those standards require that we plan and perform our examination to obtain reasonable assurance about whether management's assertion is fairly stated, in all material respects. We believe that the evidence we obtained is sufficient and appropriate to provide a reasonable basis for our opinion.

Our examination included:

- Obtaining an understanding of the system and the service organization's service commitments and system requirements
- Assessing the risks that controls were not effective to achieve GitHub's service commitments and system requirements based on the applicable trust services criteria
- Performing procedures to obtain evidence about whether controls within the system were effective to achieve GitHub's service commitments and system requirements based the applicable trust services criteria

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

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 were achieved based on the applicable trust services criteria. Also, the projection to the future of any conclusions about the 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.

Opinion

In our opinion, management's assertion that the controls within GitHub's Enterprise Cloud were effective throughout the period October 1, 2022 to September 30, 2023, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria is fairly stated, in all material respects.

Moss Adams LLP

San Francisco, California
November 15, 2023

II. GitHub's Assertion

We are responsible for designing, implementing, operating, and maintaining effective controls within GitHub's Enterprise Cloud (system) throughout the period October 1, 2022 to September 30, 2023 to provide reasonable assurance that GitHub's service commitments and system requirements relevant to Security were achieved. Our description of the boundaries of the system is presented in Section III titled "GitHub's Description of the Boundaries of Its Enterprise Cloud" and identifies the aspects of the system covered by our assertion.

We have performed an evaluation of the effectiveness of the controls within the system throughout the period October 1, 2022 to September 30, 2023, to provide reasonable assurance that GitHub's 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's objectives for the system in applying the applicable trust services criteria are embodied in its service commitments and system requirements relevant to the applicable trust services criteria. The principal service commitments and system requirements related to the applicable trust services criteria are presented in Section III titled "GitHub's Description of the Boundaries of Its Enterprise Cloud".

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 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 complementary user entity controls assumed in the design of GitHub's controls.

There are inherent limitations in any system of internal control, including the possibility of human error and the circumvention of controls. Because of these inherent limitations, a service organization may achieve reasonable, but not absolute, assurance that its service commitments and system requirements are achieved.

We assert that the controls within the system were effective throughout the period October 1, 2022 to September 30, 2023, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria.

III. GitHub's Description of the Boundaries of Its Enterprise Cloud

A. System Overview

1. 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 and Oxford, UK. 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 for software code, GitHub offers the distributed version control and source code management functionality of Git with additional features and enhancements. Specifically, it provides access control and several collaboration features including bug tracking, feature requests, task management, GitHub Advanced Security, GitHub Teams, GitHub Actions, pull requests, discussions, issues, pages, projects, docs, and wikis.

GitHub 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) 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.

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

CODE MANAGEMENT

Code review is a critical path to better code, and 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 when 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 allow for 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 Enterprise Cloud features.

PROJECT MANAGEMENT

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

Issues enable team task tracking, with resources identified and 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 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 pull-request locking, to help teams stay focused on code. For maintaining open-source projects, user blocking reduces noise and keeps conversations productive.

DOCUMENTATION

GitHub allows for 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, GitHub Advanced Security, GitHub Teams, 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.*

2. 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 might 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.
- Application 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

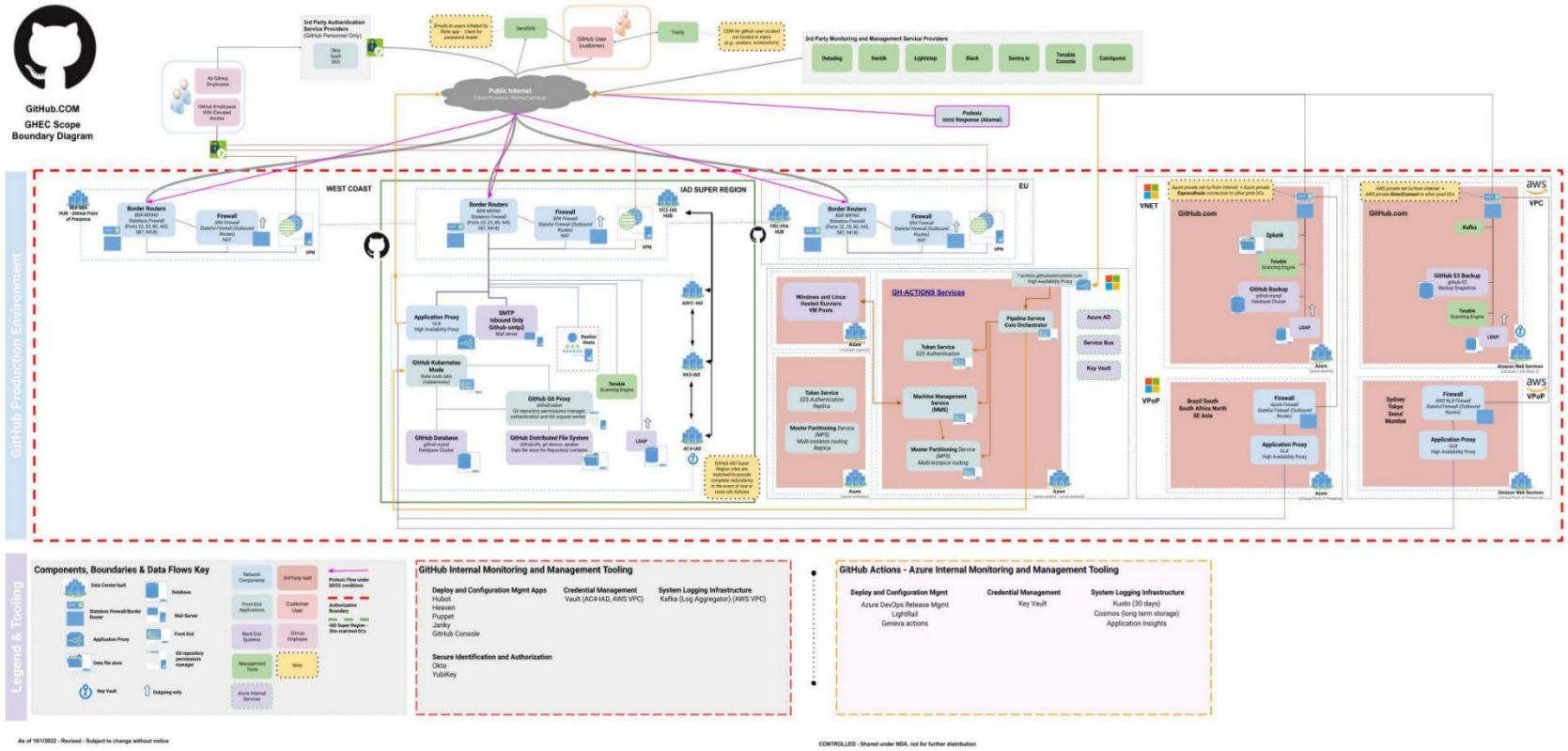
Enterprise Cloud infrastructure is hosted in geographically distributed data centers, specifically located in Virginia (Dulles and Ashburn), Washington state (Seattle and Tukwila), and Germany (Frankfurt).

Separately, backups from the Git file servers and database servers are maintained in geographically distinct AWS and Azure data center locations.

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

NETWORK ARCHITECTURE

GitHub employs the use of 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
SSO
GitHub User (customer)
3rd Party Monitoring and Management Service Providers
Datadog
Kentik
Lightstep
Slack
Sentry.io
Tenable Console
Catchpoint
Public Internet
Prolexic DDOS Response (Akamai)
GitHub Production Environment
WEST COAST
IAD SUPER REGION
EU
Border Routers
Firewall
VPN
SMTP Inbound Only Github-smtp2
GitHub Kubernetes Mode
GitHub Database
GitHub Git Proxy
GitHub Distributed File System
Tenable Scanning Engine
LDAP
GIT-ACTIONS Services
Windows and Linux Hosted Runners VM Pools
Token Service S2S Authentication
Machine Management Service (MMS)
Pipeline Service Core Orchestrator
Azure AD
Service Bus
Key Vault
Master Partitioning Service (MPS) Multi-instance routing
VNET
VPC
VPoP
GitHub.com
Splunk
GitHub Backup
GitHub S3 Backup
Kafka
Brazil South
South Africa North
SE Asia
Sydney
Tokyo
Seoul
Mumbai
Application Proxy
Legend & Tooling
Components, Boundaries & Data Flows Key
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
Lightitail
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.

3. 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 utilized 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 Ubuntu Focal, 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.

4. 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 Chief Executive Officer (CEO), Head of Finance, Chief Revenue Officer (CRO), Chief Security Officer (CSO), Chief Human Resource Officer (CHRO), Head of Design, Vice President of Strategy, Chief of Staff, Vice President, 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: Threat Hunting Operations and Response (THOR), Product Security Incident Response Team (PSIRT), Security Lab, Security Operations, Secure Access Engineering, Security Telemetry, Vulnerability Management, Cloud and Enterprise Security, and Governance, Risk, Compliance, and Communication (GRCC). These teams manage security incident detection and response, monitoring, vulnerability scanning, network and application layer penetration testing, security architecture, security engineering and operations, access management, endpoint asset management, and risk and compliance oversight.

- *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 desktops, laptops, software licenses, and other gear required by personnel, as well as fielding requests and provisioning access to corporate systems. 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, and 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 Support* – 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.

5. 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 specified. GitHub aggregates metadata and parses content patterns to deliver generalized insights within the product. It uses data from public repositories, and uses metadata and aggregate data from private repositories when a repository's owner has chosen to share the data with GitHub through an opt-in.

If a private repository is opted in for data use to take advantage of any of the capabilities of the security and analysis features, then GitHub 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, users can refer to the following link: How GitHub Uses & Protects Your Data (https://docs.github.com/en/site-policy/privacy-policies/github-privacy-statement).

6. 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 and Privacy Management System (ISPMS) Policy and related Policies define information security practices, roles, and responsibilities. The ISPMS outlines the security roles and responsibilities for the organization and expectations for employees, contractors, and third parties utilizing GitHub systems or data.

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

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

Policy	Implementing Standard(s)	Procedures
GitHub ISPMS		
GitHub ISPMS Scope		
GitHub ISPMS Statement of Applicability (SOA)		
Standards directly associated with the ISPMS		

Policy	Implementing Standard(s)	Procedures
	Chatops Command Security and Risk Standard	
	Controls Monitoring Standard	
		Controls Monitoring SOP
	Data Classification Standard	
	Domain Management Standard	
	Endpoint Security Standard	
	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		

Policy	Implementing Standard(s)	Procedures
Full Time Employee Termination Policy		
Identity and Access Management Policy		
	Identity and Access Management Standard	
		IAM Onboarding SOP
		IAM Entitlements SOP
		IAM Privileged Systems and Elevated Access SOP
		Granting Slack Access to Contractors and Consultants SOP
		IAM Non-Human Accounts in Okta
		IAM Offboarding SOP
		IAM On-Leave SOP
Physical and Environmental Protection Policy		
	Production Datacenter Standard	
		Datacenter Physical Access SOP
		Production Media Destruction SOP
		Datacenter Access compliance guidelines
Privacy Statement		
Private Information Removal Policy - External Customer Facing Policy		
Secure Coding Policy		
	Secure Coding Standard	
		Secure Coding - Dotcom

Policy	Implementing Standard(s)	Procedures
		Secure Coding - General Guidance
		Security Requirements for New Applications
Security Awareness and Privacy Training Policy		
	Security Awareness Training Standard	
Security Event Logging and Monitoring Policy		
	Security Event Logging Standard	
		Security Event Logging SOP
	Security Event Monitoring and Alerting Standard	
Security Incident Response and Data Breach Notification Policy		
	Data Breach Notification Standard	
	Security Incident Response Standard	
		Security Incident Response Procedure
		Data Breach Notification Procedure
		Security Concern Reporting Procedure
Security Policy Exception Policy		
Security Risk Management Policy		
		Security Risk Reporting - Standard Operating Procedure
		Security-GRCC Vendor Risk Assessment Process
System and Services Acquisition Policy		

Policy	Implementing Standard(s)	Procedures
	Vendor Security Standard	
		Purchasing Workflow
		Vendor Security Reviews SOP
		Procurement Workflow
		Vendor Off-boarding Checklist
		Decommissioning a GitHub-Owned App
		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		

Policy	Implementing Standard(s)	Procedures
Resilience Program Policy		
	Resiliency Standard	
Security Document Management Policy		
	Security Document Management Standard	

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. 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, taking into account the types of controls expected to be implemented by the subservice organization as described below.

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

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

D. 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 achieve certain criteria included in this report.

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	
1	Enabling SAML for their Enterprise Cloud accounts.
2	Enabling two-factor authentication and ensuring members and collaborators require two-factor authentication; this includes the implementation and management of personal access tokens.
3	Administering and configuring repositories, including permissions, enabling required reviews for pull requests, and enabling required status checks before merging.
4	Securing configuration of GitHub Actions. See https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions for detailed guidance.

MOSSADAMS