Proprietary & Confidential

GitHub

System Description of the 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, 2020 TO SEPTEMBER 30, 2021

MOSSADAMS

Table of Contents

I. Independent Service Auditor's Report 1

II. GitHub's Assertion 4

III. GitHub's Description of the Boundaries of Its Enterprise Cloud 5

A. System Overview 5

1. Services Provided 5
2. System Boundaries 7
3. Subservice Organizations 7
4. Infrastructure 7
5. Software 9
6. People 9
7. Data 10
8. Processes and Procedures 11

B. Principal Service Commitments and System Requirements 13

C. Complementary Subservice Organization Controls 14

D. Complementary User Entity Controls 15

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, 2020 to September 30, 2021, 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 multiple subservice organizations for colocation data center services and 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 organization, 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 have complied with the independence and other ethical requirements of the Code of Professional Conduct established by the AICPA.

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, 2020 to September 30, 2021, 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 12, 2021

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, 2020 to September 30, 2021 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 entitled "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, 2020 to September 30, 2021, 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 entitled "GitHub's Description of the Boundaries of Its Enterprise Cloud".

GitHub uses multiple subservice organizations for colocation data center services and 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.

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, 2020 to September 30, 2021, 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; Raleigh, NC; Oxford, UK; Tokyo, Japan; Hyderabad, India; and Amsterdam, Netherlands. GitHub currently employs approximately 2,100 employees, with approximately 70 percent of the workforce 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, and wikis.

GitHub Enterprise Cloud is GitHub's SaaS solution for collaborative software development. Features of the Enterprise Cloud service organizations, code hosting, code management, project management, team management, and documentation:

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, attached files, and link directly to the latest download.

CODE MANAGEMENT

Code review is 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.

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.

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.

2. System Boundaries

The scope of this report includes the GitHub Enterprise Cloud, and the supporting production systems, infrastructure, software, people, procedures, and data. GitHub offers both hosted and customer on-premise versions. However, the GitHub Enterprise On-premise, Jobs, and Pages services are excluded from the scope of this report.

3. 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 Amazon Web Services (AWS) and Azure 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.

4. 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.
- API front-end servers are the interface to any client using the Advanced Programming Interface (API) to interact with GitHub programmatically.
- 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.

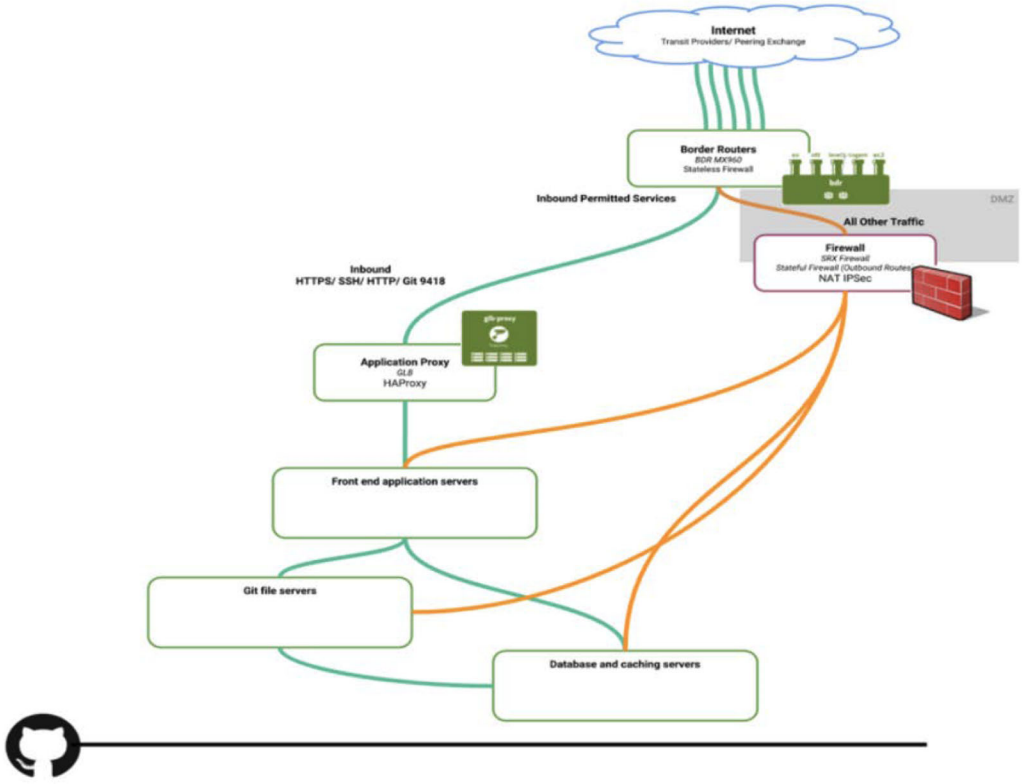
DATA CENTERS

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

Separately, backups from the Git file servers and database servers are maintained in geographically distinct AWS Simple Storage Solution (S3) data center locations.

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.



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.

5. Software

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 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 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.

6. 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, CFO, COO, CSO, Chief Human Resources Officer, Chief of Staff, Senior Vice President of Product, Senior Vice President of Technology, VP of Worldwide Sales, General Manager over Azure DevOps, VP of Strategy, VP of Communications, VP of Microsoft Partnership, and General Counsel.
- *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: Security Incident Response Team (SIRT), Product and Application Security, Security Operations Vulnerability Management Engineering, and Compliance and Risk Management. 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. The Site Reliability Engineering (SRE) team conducts post-mortem reviews of emergency deploys.
- *Information Technology (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, ensuring compliance with internal contractual standards, and maintaining data privacy policies and standards.
- *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.

7. Data

GitHub uses repository data to connect users to relevant tools, people, projects, and information. Repositories are categorized as either public or private. 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 the repository owner opts a private repository into data use, then it will perform read-only analysis of that specific private repository.

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 Secure Sockets Layer (SSL). 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 data, refer to the following link: How Github Uses & Protects Your Data https://docs.github.com/en/free-pro-team@latest/github/understanding-how-github-uses-and-protects-your-data/about-githubs-use-of-your-data.

8. 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 are updated and approved as necessary for changes in the business, at a minimum annually.

GitHub policies establish procedures and controls to enable security, efficiency, availability, and quality of service. The GitHub Entity Security Policy and related policy statements define information security practices, roles, and responsibilities. The Entity Security Policy 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, referenced therein. 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, standards, and procedures operating in support of the Entity Security Policy objectives:

Document	Type
GitHub Identity and Access Management Policy Statement	Policy
Vulnerability Management Policy Statement	Policy
Security Risk Management Policy Statement	Policy
GitHub Corporate Data Retention Policy Statement	Policy
GitHub Security Incident Response and Data Breach Notification Policy	Policy
Policy Exception Policy Statement	Policy
GitHub Production Data Center Access Policy Statement	Policy
Security and Privacy Awareness Training Policy Statement	Policy
Change Management Policy Statement - How Hubbers Build Software	Policy

Document	Type
System and Services Acquisition Policy Statement	Policy
Background Check Policy	Policy
Secure Coding Policy Statement	Policy
Identity and Access Management Security Standards	Standard
Vendor Security Standards	Standard
Patch Management Standard	Standard
Chatops Command Security and Risk Standard	Standard
GitHub Data Classification Standard	Standard
GitHub Security Event Logging Standard	Standard
High Risk Repo Administration	Standard
Encryption Standard	Standard
Secure Coding Principles	Standard
Operating System Hardening Standard	Standard
Database Hardening Standard	Standard
Disaster Recovery Standard	Standard
Control Monitoring SOP	Procedure
Decommissioning a GitHub-Owned App	Procedure
Security Event Logging SOP	Procedure
GitHub Security Risk Reporting SOP	Procedure

Document	Type
GitHub Security Incident Response Plan	Procedure
GitHub Data Breach Notification Plan	Procedure
GitHub Inventory SOP	Procedure
GitHub Production Media Destruction SOP	Procedure
IAM onboarding SOP	Procedure
IAM offboarding SOP	Procedure
IAM - Slack Access to Contractors and Consultants SOP	Procedure
Vendor Security Reviews SOP	Procedure
Vulnerability Management Process	Procedure
Security and Privacy Awareness Training SOP	Procedure

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 management has determined that complementary controls at its subservice organizations that are suitably designed and operating effectively are necessary, along with controls at GitHub, to achieve GitHub's service commitments and system requirements related to the Enterprise Cloud, based on the applicable trust services criteria. Therefore, each user entity's internal controls should be evaluated in conjunction with GitHub's controls and the related tests and results described in Section IV of this report, while also taking into account the related complementary subservice organization controls expected to be implemented at the subservice organizations as described below.

	Complementary Subservice Organization Controls	Related Criteria	Relevant Subservice Provider(s)
1	Access to hosted systems requires strong authentication mechanisms.	**CC 6.1**	AWS, Azure, Sabey, QTS, Coresite, Equinix
2	New and existing user access and permissions to hosted systems are approved by appropriate personnel prior to be granted.	**CC 6.1, CC 6.2, CC 6.3**	AWS, Azure, Sabey, QTS, Coresite, Equinix
3	Terminated user access permissions to hosted systems are removed in a timely manner.	**CC 6.1, CC 6.2, CC 6.3**	AWS, Azure, Sabey, QTS, Coresite, Equinix
4	User access permissions to hosted systems are reviewed by appropriate personnel on a regular basis.	**CC 6.2, CC 6.3**	AWS, Azure, Sabey, QTS, Coresite, Equinix
5	Privileged access to hosted systems and the underlying data is restricted to appropriate users.	**CC 6.3, CC 6.7**	AWS, Azure, Sabey, QTS, Coresite, Equinix
6	Access to the physical facilities housing hosted systems is restricted to authorized users.	**CC 6.4**	AWS, Azure, Sabey, QTS, Coresite, Equinix
7	Production media is securely decommissioned and physically destroyed prior to being removed from the data center.	**CC 6.5**	AWS, Azure, Sabey, QTS, Coresite, Equinix
8	Network security mechanisms restrict external access to the production environment to authorized ports and protocols.	**CC 6.6**	AWS, Azure, Sabey, QTS, Coresite, Equinix
9	Connections to the production environment require encrypted communications.	**CC 6.6, CC 6.7**	AWS, Azure, Sabey, QTS, Coresite, Equinix
10	Antivirus or antimalware solutions detect or prevent unauthorized or malicious software on hosted systems.	**CC 6.8**	AWS, Azure, Sabey, QTS, Coresite, Equinix

	Complementary Subservice Organization Controls	Related Criteria	Relevant Subservice Provider(s)
11	System configuration changes are enforced, logged, and monitored.	**CC 6.8, CC 7.1**	AWS, Azure, Sabey, QTS, Coresite, Equinix
12	Hosted systems are scanned for vulnerabilities. Any identified vulnerabilities are tracked to resolution.	**CC 7.1**	AWS, Azure, Sabey, QTS, Coresite, Equinix
13	System activities on hosted systems are logged, monitored and evaluated for security events. Any identified incidents are contained, remediated and communicated according to defined protocols.	**CC 7.2, CC 7.3, CC 7.4**	AWS, Azure, Sabey, QTS, Coresite, Equinix
14	Access to make changes to hosted systems is restricted to appropriate personnel.	**CC 8.1**	AWS, Azure, Sabey, QTS, Coresite, Equinix
15	Changes to hosted systems are documented, tested, and approved prior to migration to production.	**CC 8.1**	AWS, Azure, Sabey, QTS, Coresite, Equinix

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 customer organizations 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 customer organizations to complement the controls at GitHub. User auditors should consider whether the following controls have been placed in operation by the customers.

Each customer must evaluate its own internal control structure to determine if the identified customer controls are in place. Users are responsible for:

	Complementary User Entity Controls	Related Criteria
1	Enabling SAML for their Enterprise Cloud accounts.	**CC 6.1**
2	Enabling two-factor authentication and ensuring members and collaborators require two-factor authentication.	**CC 6.1 and CC 6.6**
3	Creating and managing their Organization and Teams, including the proper configuration of access permissions to repositories.	**CC 6.2 and CC 6.3**

	Complementary User Entity Controls	Related Criteria
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.	**CC 6.2 and CC 6.3**
5	Ensuring authorized users are appointed as Organization owners for administration of the Organization.	**CC 6.2 and CC 6.3**
6	Maintaining an effective onboarding and offboarding process for their own employees and contractors.	**CC 6.2 and CC 6.3**
7	Reviewing events in the security logs.	**CC 7.1**
8	Administering and configuring repositories, including permissions, enabling required reviews for pull requests, and enabling required status checks before merging.	**CC 8.1**
9	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.	**CC 9.2**

MOSSADAMS