COALFIRE
CONTROLS

Report on GitHub, Inc.'s Description of Its "GitHub Copilot Business" Service and on the Suitability of the Design of Its Controls Relevant to Security as of April 30, 2024

SOC 2® - SOC for Service Organizations: Trust Services Criteria

Table of Contents

Section 1

Section 2

Section 3

Section 4

Section 1

Independent Service Auditor's Report

Independent Service Auditor’s Report

To: GitHub, Inc. (“GitHub”)

Scope

We have examined GitHub’s accompanying description of its GitHub Copilot Business service found in Section 3 titled “GitHub, Inc.’s Description of Its GitHub Copilot Business service as of April 30, 2024” (description), based on the criteria for a description of a service organization’s system set forth in DC Section 200, *2018 Description Criteria for a Description of a Service Organization’s System in a SOC 2® Report (With Revised Implementation Guidance—2022*) (2018 description criteria), and the suitability of the design of controls stated in the description as of April 30, 2024, to provide reasonable assurance that GitHub’s service commitments and system requirements would be 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 (With Revised Points of Focus—2022)* (2017 TSC).

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.

GitHub uses subservice organizations to provide data center colocation services. 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.

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 would be achieved. In Section 2, GitHub has provided the accompanying assertion titled “Assertion of GitHub, Inc. Management” (assertion) about the description and the suitability of the design 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 design of 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. 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 to provide reasonable assurance that the service organization’s service commitments and system requirements would be 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.

We are required to be independent and to meet our other ethical responsibilities in accordance with relevant ethical requirements relating to the examination engagement.

An examination of a description of a service organization’s system and the suitability of the design of controls involves—

- Obtaining an understanding of the system and the service organization’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.
- Performing procedures to obtain evidence about whether the description is presented in accordance with the description criteria.
- Performing procedures to obtain evidence about whether controls stated in the description were suitably designed to provide reasonable assurance that the service organization would achieve 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.

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 report users may consider important to meet their informational needs. There are inherent limitations in any system of internal control, including the possibility of human error and the circumvention of controls. The projection to the future of any conclusions about the suitability of the design 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.

Other Matter

We did not perform any procedures regarding the operating effectiveness of controls stated in the description and, accordingly, do not express an opinion thereon.

Opinion

In our opinion, in all material respects—

a. The description presents the GitHub Copilot Business service that was designed and implemented as of April 30, 2024, in accordance with the description criteria.

b. The controls stated in the description were suitably designed as of April 30, 2024, to provide reasonable assurance that GitHub's service commitments and system requirements would be achieved based on the applicable trust services criteria, if its controls operated effectively as of that date and if the subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls as of that date.

Restricted Use

This report is intended solely for the information and use of GitHub, user entities of the GitHub Copilot Business service as of April 30, 2024, business partners of GitHub subject to risks arising from interactions with the GitHub Copilot Business service, 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. If a report recipient is not a specified party as defined above and has obtained this report, or has access to it, use of this report is the non-specified user's sole responsibility and at the non-specified user's sole and exclusive risk. Non-specified users may not rely on this report and do not acquire any rights against Coalfire Controls, LLC as a result of such access. Further, Coalfire Controls, LLC does not assume any duties or obligations to any non-specified user who obtains this report and/or has access to it.

Coalfire Controls LLC

Greenwood Village, Colorado
May 16, 2024

Section 2

Assertion of GitHub, Inc. Management

GitHub

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

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

We have prepared the accompanying description of the GitHub Copilot Business service titled “GitHub, Inc.’s Description of Its GitHub Copilot Business service as of April 30, 2024” (description), based on the criteria for a description of a service organization’s system set forth in DC Section 200, *2018 Description Criteria for a Description of a Service Organization’s System in a SOC 2® Report (With Revised Implementation Guidance—2022)* (2018 description criteria). The description is intended to provide report users with information about the GitHub Copilot Business service that may be useful when assessing the risks arising from interactions with GitHub’s system, 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 (With Revised Points of Focus—2022)* (2017 TSC).

GitHub uses subservice organizations for data center colocation services. 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:

a. The description presents the GitHub Copilot Business service that was designed and implemented as of April 30, 2024, in accordance with the description criteria.

b. The controls stated in the description were suitably designed as of April 30, 2024, to provide reasonable assurance that GitHub’s service commitments and system requirements would be achieved based on the applicable trust services criteria, if its controls operated effectively as of that date, and if the subservice organizations and user entities applied the complementary controls assumed in the design of GitHub’s controls as of that date.

GitHub, Inc.

Section 3

GitHub, Inc.'s Description of Its GitHub Copilot Business service as of April 30, 2024

Type of Services Provided

GitHub ("the Company") is an independently operated subsidiary of Microsoft, and 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.

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.

GitHub Copilot is an Artificial Intelligence (AI)-powered coding assistant that helps developers write code faster. GitHub Copilot is available through GitHub personal accounts with GitHub Copilot Individual or through organization or enterprise accounts with GitHub Copilot Business.

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

Principal Service Commitments and System Requirements

Commitments are declarations made by management to customers regarding the performance of GitHub Copilot Business. System requirements are the specifications about how the Copilot business service should function. System requirements should:

- Support Service Commitments: Ensure it functions in a way that fulfills promises made to users, their customers, vendors, and business partners.
- Follow Rules and Regulations: Operate within all relevant laws and any guidelines set by industry groups.
- Meet Additional Goals: Help the organization achieve other objectives important to building trust with those it serves.

The Company's principal service commitments and system requirements related to GitHub Copilot Business include the following:

Trust Services Category	Service Commitments
Security	• The GitHub leadership team will continually establish reasonable and appropriate security measures, along with identifying and mitigating risks to the entity and customers. • The GitHub leadership team is committed to building a culture of security within GitHub and the software development and open source communities. • GitHub attracts, develops, and attempts to retain competent individuals in all areas of the organization to accomplish effective technical and human security measures. • GitHub designs reasonable and appropriate security controls to effectively mitigate risks and conducts periodic internal and external assessments to analyze the design and operating effectiveness of the security controls. Where control design or operational flaws are identified, GitHub works to remediate those flaws in a timely manner. • GitHub will utilize reasonable and appropriate security measures related to configuration and software development change management, highlighted specifically by the core use cases and features directly built into the GitHub platform. • GitHub will utilize reasonable and appropriate security and legal measures to manage vendor and business partner risks. • GitHub will utilize reasonable and appropriate security measures to safeguard sensitive information against unauthorized access, use, modification, destruction, or disclosure. • If GitHub becomes aware of a security incident, GitHub will notify the customer without undue delay. • GitHub Copilot Business communications will be encrypted in-transit between GitHub and the customer, using Transport Layer Security (TLS) v1.2 and above with HTTP Strict Transport Security (HSTS) enabled. • As documented in the Copilot Product Terms, Prompts will not be retained by GitHub when using Copilot Business Chat and Code Completion in the Interactive Development Environment (IDE). • Copilot Business will not retain Suggestions produced by Copilot when using Copilot Business Chat and Code Completion in the IDE. • Copilot Business will not retain user content (e.g. code from the user's editor) when using Copilot Business Chat and Code Completion in the IDE. • As documented in the Copilot Product Terms, GitHub will allow customer administrators to enable the Duplicate Detection feature to filter suggestions that match existing public code. • Copilot Business does not use the customer's code to improve the model. • Copilot Business does not send information to OpenAI, the software company.

The Company designs its processes and procedures to provide a secure environment for customer data. GitHub's security commitments and system requirements are documented and communicated to customers in the Terms of Service and at the 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://github.com/customer-terms

The Components of the System Used to Provide the Services

The boundaries of GitHub Copilot Business are the specific aspects of the Company’s infrastructure, software, people, procedures, and data necessary to provide its services and that directly support the services provided to customers. Any infrastructure, software, people, procedures, and data that indirectly support the services provided to customers are not included within the boundaries of GitHub Copilot Business.

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

Infrastructure

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

The in-scope hosted infrastructure also consists of multiple supporting tools, as shown in the table below:

Infrastructure			
Production Tool	**Business Function**	**Operating System**	**Hosted Location**
Border/Edge Routers	GitHub load balancers, application layer proxies, and firewalls are the systems that connect to the internet. These routers, application proxies, and firewalls are the first line of defense in protecting the system.	Juniper JUNOS / Azure VMs	Azure and GitHub Colocation Data Centers
GitHub Copilot Business API	Application programming interface (API) front-end servers are the interface to GitHub Copilot Business Chat clients	Ubuntu	Azure and GitHub Colocation Data Centers
GitHub Copilot Business Proxy	Service that manages connections between Code Completion plugins and AI models	Ubuntu	Azure and GitHub Colocation Data Centers
GitHub Enterprise Cloud	GitHub Copilot Business authentication and license management	Ubuntu, Mysql	Azure and GitHub Colocation Data Centers

Software

Software consists of the programs and software that support GitHub Copilot Business (operating systems [OSs], middleware, and utilities). The list of software and ancillary software used to build, support, secure, maintain, and monitor GitHub Copilot Business include the following applications, as shown in the table below:

Software	
Production Application	**Business Function**
Azure OpenAI	AI models
Datadog	Application and infrastructure monitoring
GitHub Actions	Continuous Integration / Continuous Deployment (CI/CD)
GitHub Advanced Security	Dependency Management, Static Code Analysis, Secret Scanning
GitHub Enterprise Cloud	Source Code Management, authentication, license management
Jamf / Defender	Endpoint security
MySQL	Account and license storage
Puppet	Infrastructure Configuration Management
Splunk	Security information and event management (SIEM), logging system, intrusion detection
Terraform	Infrastructure Configuration Management
Ubuntu	OS Baseline
ZenDesk	Customer Support

People

The Company develops, manages, and secures GitHub Copilot Business via separate departments. The responsibilities of these departments are defined in the following table:

People	
Group/Role Name	**Function**
Customer Support	Responsible for providing technical and account-related support to GitHub Copilot Business customers and for resolving customer issues via email, chat, social media, and phone from developers and customer entities around the globe.

People	
Group/Role Name	**Function**
Engineering	Responsible for working with the Product team to plan and coordinate releases, and accountable for building, testing, and deploying GitHub Copilot Business code and feature changes. Responsible for maintaining service availability, including performance and scale monitoring and reporting, incident command, and on-call readiness for any production issues. Responsible for configuration management, building, testing, and deploying software relevant to the operation and management of production assets, patching and remediation of vulnerabilities reported by the Security team, and data center operations management. Responsible for managing Git and database storage backups and restores.
Legal	Responsible for negotiating contractual obligations with third parties and technology partners/suppliers, legal terms and conditions, and ensuring compliance with internal contractual standards.
People Operations	Responsible for talent acquisition, diversity and inclusion, learning and development, and employee engagement on everything from benefits and perks to career development and growth.
Privacy	Responsible for determining which privacy laws and regulations apply to GitHub and determining the best way to comply with them, ultimately ensuring GitHub can offer its products to every developer anywhere in the world.
Product Management	Responsible for understanding customer requirements, collecting, defining, and clarifying feature requests and development efforts, and managing feature rollouts and related customer communication efforts.
Security	Responsible for ensuring the security of GitHub products. Security consists of multiple teams with specific missions: Threat Hunting Operations and Response Incident Response (THORIR), 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.
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, Chief Operations Officer, Chief of Staff, Vice President, Senior Vice President of Engineering, Chief Legal Officer, and Vice President of Communities.

Policies, Standards, and Procedures

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

GitHub Policies and Standards establish controls to enable security, efficiency, availability, and quality of service. The GitHub Information Security 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.

The following table details the procedures as they relate to the operation of GitHub Copilot Business:

Procedures	
Policy	**Associated Standards and Procedures**
GitHub ISPMS	No Associated Standards or Procedures
GitHub ISPMS Scope	No Associated Standards or Procedures
GitHub ISPMS Statement of Applicability (SOA)	• 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 • High-Risk Application Access Standard • Organization Administration Standard • Production VPN Access Standard • Repository Security Baseline Configuration Standard • Reviewing Pull Requests • Server Operating System Standard
Corporate Data Retention Policy	• Audit Video Retention Standard • Corporate Data Retention Standard • Product Telemetry Data Retention Standard • Slack Retention Standard
Contractor Termination Policy	No Associated Standards or Procedures
Full Time Employee Termination Policy	No Associated Standards or Procedures

Procedures	
Policy	**Associated Standards and Procedures**
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	No Associated Standards or Procedures
Private Information Removal Policy - External Customer Facing Policy	No Associated Standards or Procedures
Secure Coding Policy	Secure Coding Standard • Secure Coding - Dotcom • Secure Coding - General Guidance • Security Requirements for New Applications
Security Awareness and Privacy Training Policy	Security Awareness Training Standard
Security Event Logging and Monitoring Policy	• Security Event Logging Standard • Security Event Logging SOP • Security Event Monitoring and Alerting Standard
Security Incident Response and Data Breach Notification Policy	• Data Breach Notification Standard • Security Incident Response Standard • Security Incident Response Procedure • Data Breach Notification Procedure • Security Concern Reporting Procedure
Security Policy Exception Policy	No Associated Standards or Procedures
Security Risk Management Policy	• Security-GRCC Vendor Risk Assessment Process • Security Risk Reporting - Standard Operating Procedure
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 • Vendor Offboarding SOP • Encryption Standard

Procedures	
Policy	**Associated Standards and Procedures**
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	No Associated Standards or Procedures
IT Asset Management Policy	No Associated Standards or Procedures
Network Policy	No Associated Standards or Procedures
Resilience Program Policy	Resiliency Standard
Security Document Management Policy	Security Document Management Standard

Data

Data refers to transaction streams, files, data stores, tables, and output used or processed by the Company. Customers interact with GitHub Copilot Business by providing prompts from editor plugins, and optionally from the mobile app, the command-line interface (CLI), or Chat surfaces in GitHub.com. GitHub Copilot Business returns a suggestion based on the prompt. GitHub Copilot Business does not retain prompts or suggestions when used from editor plugins. GitHub Copilot Business may retain prompts and suggestions for up to 28 days when optionally used with the mobile App, the CLI or Chat surfaces in GitHub.com.

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

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

The following table details the types of data contained in the production application for GitHub Copilot Business:

Data		
Production Application	**Description**	**Data Store**
Authentication / License Data	GitHub Enterprise Cloud holds user authentication data and license information	GitHub Enterprise Cloud

Data		
Production Application	**Description**	**Data Store**
Prompts	“Prompt” means the collection of code and supporting contextual information that GitHub Copilot Business sends to GitHub to generate Suggestions. This also includes the content that you submit through a chat interface.	Not Stored when using GitHub Copilot Business Chat and Code Completion in the IDE. Stored in CosmosDB when using the optional CLI or Mobile app, or Chat in GitHub.com.
Suggestions	“Suggestions” means the code, functions, and other output returned to you by GitHub Copilot Business.	Not Stored when using GitHub Copilot Business Chat and Code Completion in the IDE. Stored in CosmosDB when using the optional CLI or Mobile app, or Chat in GitHub.com.
User Engagement Data	User engagement data is usage information about events generated when interacting with a code editor. These events include user edit actions (for example: whether a suggestion was accepted or dismissed, but not the content of the suggestion), error messages, and general usage data to identify user metrics such as latency and feature engagement. This information may include personal data, such as pseudonymous identifiers.	GitHub Copilot Business Telemetry Service

System Incidents

There were no identified significant system incidents that (a) were the result of controls that were not suitably designed or operating effectively to achieve one or more of the service commitments and system requirements or (b) otherwise resulted in a significant failure in the achievement of one or more of those service commitments and system requirements as of April 30, 2024.

The Applicable Trust Services Criteria and Related Controls

Applicable Trust Services Criteria

The Trust Services Category that is in scope for the purposes of this report is:

- *Security*: Information and systems are protected against unauthorized access, unauthorized disclosure of information, and damage to systems that could compromise the information or systems and affect the entity's ability to meet its objectives.

The security criteria are organized as follows:

1. *Control environment:* The criteria relevant to how the entity is structured and the processes the entity has implemented to manage and support people within its operating units. This includes criteria addressing accountability, integrity, ethical values, qualifications of personnel, and the environment in which they function.
2. *Information and communication:* The criteria relevant to how the entity communicates its policies, processes, procedures, commitments, and requirements to authorized users and other parties of the system and the obligations of those parties and users to the effective operation of the system.
3. *Risk assessment:* The criteria relevant to how the entity (i) identifies potential risks that would affect the entity's ability to achieve its objectives, (ii) analyzes those risks, (iii) develops responses to those risks including the design and implementation of controls and other risk mitigating actions, and (iv) conducts ongoing monitoring of risks and the risk management process.
4. *Monitoring activities:* The criteria relevant to how the entity monitors the system, including the suitability and design and operating effectiveness of the controls, and acts to address deficiencies identified.
5. *Control activities:* The criteria relevant to the actions established through policies and procedures that help ensure that management's directives to mitigate risks to the achievement of objectives are carried out.
6. *Logical and physical access controls:* The criteria relevant to how the entity restricts logical and physical access, provides and removes that access, and prevents unauthorized access.
7. *System operations:* The criteria relevant to how the entity manages the operation of system(s) and detects and mitigates processing deviations, including logical and physical security deviations.
8. *Change management:* The criteria relevant to how the entity identifies the need for changes, makes the changes using a controlled change management process, and prevents unauthorized changes from being made.
9. *Risk mitigation:* The criteria relevant to how the entity identifies, selects, and develops risk mitigation activities arising from potential business disruptions and the use of vendors and business partners.

This report is focused solely on the Security category. The Company has elected to exclude the Availability, Processing Integrity, Confidentiality, and Privacy categories.

Control Environment

Integrity and Ethical Values

The internal control environment reflects the overall attitude, awareness, and actions of executive management and other stakeholders concerning the importance of controls and the emphasis given to controls in 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.

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. An annual Standard of Business Conduct training occurs that is mandatory for all employees and contractors. This training covers key policies, describes how to report issues, and emphasizes the importance of ethical behavior. Each employee and contractor must attest to the completion of the training. Any ethical issue reported is investigated and appropriate action is taken up to the termination of employment.

Board of Directors

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

Organizational Structure

GitHub’s organizational structure provides the framework within which its activities for achieving company-wide objectives and key results are defined, planned, sponsored, executed, controlled, and monitored. Management believes that establishing a relevant organizational structure includes considering key areas of authority and responsibility and lines of reporting. Senior leadership sets company-wide objectives and key results and also reports on 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 responsibilities across the company. The organizational chart is available for employees both on GitHub’s intranet and internal human resource information system (HRIS) and is updated through automation. Management continues to evaluate its organizational structure and makes changes as necessary. Hubber responsibilities are communicated through documented job descriptions which are maintained by GitHub.

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

When leadership changes occur within the organization, the Security team is notified 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 of 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’s Philosophy and Operating Style

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

Authority and Responsibility

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

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

Non-employees, including Contingent Workers, Contractors, and Vendors are accountable for understanding and upholding the contractual obligations to GitHub for data protection, and to comply with Management oversight provided to ensure understanding of those obligations.

The Chief Security Officer (CSO) is responsible for the overall security posture, programs, and capabilities within GitHub. The CSO works with members of GitHub's leadership team and management, product management and engineering, business and system owners, and users to develop and implement prudent security services, policies, procedures, and controls, subject to the approval of leadership at GitHub.

The Deputy Chief Security officer reports to the CSO and is responsible for many of the day-to-day operations of the Security organizations and acts in the CSO's place when they are unavailable or as appropriately delegated.

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

- Ensuring security policies, standards, and procedures are in place and understood by Hubbers.
- Providing basic security direction and support for all systems and users.
- Advising software and product design and engineering, corporate systems development, and business owners in the implementation of security controls from the point of system design, through testing, implementation, and decommissioning.
- Educating employees at onboarding about security controls and processes relevant to GitHub.
- Providing on-going employee security skills and awareness education.

- Performing and facilitating product security audits.
- Reporting regularly to GitHub leadership on GitHub's status regarding information security.

Human Resources

GitHub People Operations is responsible for the GitHub employee lifecycle.

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

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 or clarify any discrepancies. If GitHub does not hire a candidate based on the results of a background check, GitHub provides the candidate with an adverse-action notice and informs them of their rights to see the information reported and to correct inaccurate information.

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

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

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

Information and Communication

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

Internal Communications:

GitHub has published policies and procedures, both included in the 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. Policies and Standards are available on The Hub, GitHub's centralized internal communications platform.

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

Business Conduct, instructions for reporting ethics and integrity concerns to Microsoft (including anonymous channels), and details of GitHub's policies against retaliation for reporting concerns.

External Communications:

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

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

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

Risk Assessment and Mitigation

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

The Security GRCC team is embedded within the larger Security team. This team monitors GitHub's internal controls and conducts risk monitoring in accordance with relevant regulatory and contractual compliance requirements. This responsibility includes managing the design, implementation, and monitoring of the GitHub Copilot Business 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 maintains an inventory of approved vendors, and reviews vendor security risk assessments every two years, or upon expansion or changes to the contracted service offering. Vendors deemed high risk, such as data center providers or other vendors storing or processing data in scope for GitHub’s regulatory or contractual requirements, undergo reassessment annually.

Risk Identification and Treatment

GitHub has defined risk management processes to identify and manage risks that may affect the system’s security. At least annually, 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 GitHub repository where the risks and mitigation actions are tracked to resolution.

Risk Reporting

Summary reporting on security risk areas is included in annual leadership reporting as part of the annual security risk assessment. GitHub's Senior Leadership team as well as extended leadership teams across Security, Engineering, 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.

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 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 service-level agreement (SLA) are handled via the documented exception process.

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

Patching occurs to address security fixes, bug-fixes, 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.

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.

Bug Bounty

In addition to third-party penetration testing, GitHub participates in HackerOne's Bug Bounty Program to supplement the penetration testing activities. The Bug Bounty team manages the GitHub Security Bug Bounty program. 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.

Monitoring

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 to the Senior Leadership team for awareness on a quarterly basis. Additionally, an “Internal” ISO audit is conducted each year; because GitHub does not have its own independent audit function, this audit is performed by a certified ISO audit firm, and results are reported to management in accordance with audit processes.

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

Control Activities

Logical Access

GitHub Copilot Business Employee Access

GitHub Copilot Business is developed using GitHub Enterprise Cloud. GitHub has implemented access protection measures to only allow authenticated and authorized users access to the portions of GitHub's instance of Enterprise Cloud that are not explicitly public. Access to GitHub's instance of GitHub 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 phishing-resistant authenticators.

Access to internal systems is restricted through unique account IDs ("handles"), password, and 2FA. To access these environments, GitHub users leverage the same GitHub handles as 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 GitHub Copilot Business production systems is restricted to authorized personnel with demonstrated need and isolated from the Internet by virtual private network (VPN), bastion hosts, and/or a security assertion markup language (SAML) solution enforcing Hypertext Transfer Protocol Secure (HTTPS) reverse proxy. GitHub uses secure shell protocol (SSH) to authenticate to back-end production resources through a bastion host. VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.

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

GitHub Copilot Business Customer Access

Authentication to GitHub Copilot Business requires a unique account and password. If elected, customers can also enable 2FA for their organization through Enterprise Cloud settings, or integrate with their Identity Provider via Single Sign On. Audit logging is also available to allow organization administrators to quickly review actions performed by users. This log includes details such as who performed the action, what the action was, and when it was performed. When a customer signs in to GitHub Copilot Business, GitHub Enterprise Cloud checks whether a user has a valid GitHub Copilot Business License before granting access.

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 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 PSIRT's monitoring, detection, and response program. As needed, based on the frequency and type of activity detected, the PSIRT identifies process improvements to reduce human interactions directly with production servers.

Encryption

GitHub Copilot Business traffic is encrypted in transit over the public internet. Transmissions via TLS require a minimum of 2048-bit certificate keys, and GitHub uses TLSv1.2 and above to encrypt data during transit.

Internal User Provisioning

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

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

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

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

THOR builds automated monitors of user access events to identify anomalous user access activity. The PSIRT team investigates detections, and if PSIRT 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. An offboarding notification to Security Operations is pulled from Workday, where HR updates employee termination dates, for both planned and unplanned exits. The Security Operations team monitors the automation system for errors and can manually trigger offboardings when required.

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

System Operations

System Monitoring

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:

GitHub Copilot Business – 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	Confirmed successful phishing attempt against Hubber
High	24 Hours	Human Response within One Day	High Confidence IOC Match or Behavioral Detection
Medium	1-7 Days	Automated Response or Human Response Within a Week	Low Confidence IOC Match
Low	N/A	Automated Response or Contextual Only	Uptick in Password Changes from 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 PSIRT teams triage, investigate, and respond to a variety of security events reported by internal and external sources, including:

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

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

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

The THOR and PSIRT 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.

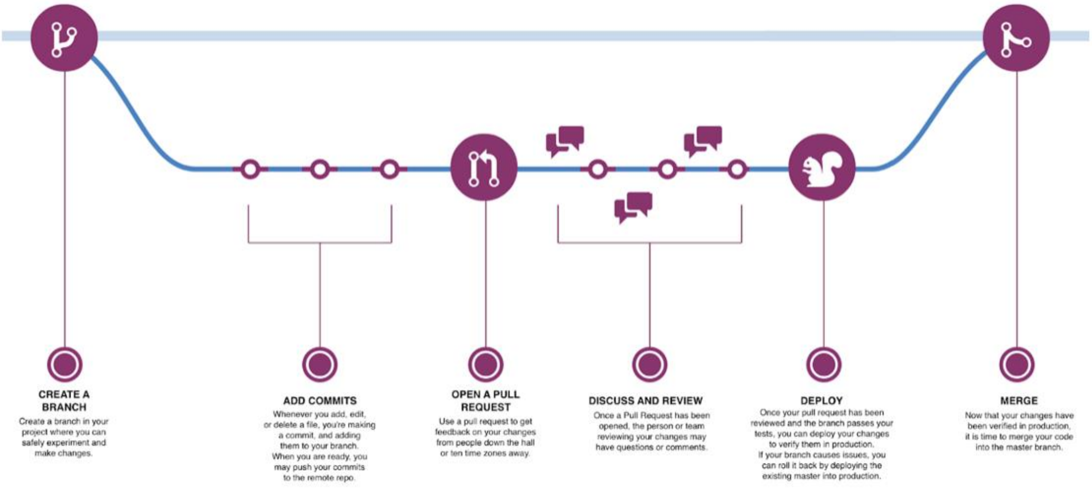
Change Management

Change Management Overview

GitHub is a collaborative, organic, and adaptable organization. GitHub Copilot Business'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 GitHub repositories is restricted to members of the Platform Health and Production Engineering teams. Engineers responsible for the GitHub Copilot Business Repositories of the codebase receive timely notifications of changes pushed and monitor these changes through automated issues posted in the 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 GitHub Copilot Business, engineers create a branch from the Main Branch, which is the 'gold copy' of GitHub, to begin development of changes. Depending on the change, developers will test locally and using automated CI/CD testing described below. Once pre-production testing has validated the change is functioning as expected and all required reviews and approvals are complete, the branch is merged into the Main Branch and is rolled out in staged releases via Canary deployments to GitHub's Kubernetes clusters in production for customer use.

An automated static code analysis tool runs every time new code is committed, to detect insecure coding practices based on third party provided lexical, syntactic, and semantic rulesets, in addition to GitHub-developed rulesets and test scenarios. An automated alerting issue is posted on the containing issue and in a tracking repository maintained by the Product Security Engineering team, to alert them of any potentially insecure coding risks to product development. The developer assesses the potential vulnerabilities, and a peer reviewer follows up on the items during change review and approval.

Peer Review

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

Protected branches (branches where merging to the Main branch is gated by required review and approval activities) are also enabled on GitHub Copilot Business code repositories. Protected branches require that PRs are reviewed and approved by an individual who is different than proposer of the change prior to merging the code to the Main Branch.

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

Automated Testing

CI is the process by which code changes are automatically tested and validated. When a change is committed to a branch with an associated 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 repo 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, and the implementation of blocking acceptance tests. These acceptance tests typically function at the browser level and focus on the overall functionality of the module or application to help ensure customer commitments and system requirements are met. GitHub prevents any change from deployment to production if required acceptance tests have not passed.

Deployment

To support deployment to the production environment, changes are deployed using GitHub's standard and proprietary deployment tools and workflows. To deploy any application or configuration changes, developers run a Slack 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. Automated deployment processes validate that required checks are completed, including passing of mandatory acceptance checks and ensuring the most recent Main Branch build is used, before deploying the change to production. Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification.

Monitoring, Metrics, and Change Rollback

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

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

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

To keep GitHub personnel in the loop on new feature development or other engineering efforts, the Product teams regularly host internal demos for the organization as a part of regular company and technology all-company 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 blog posts and may also include social posts and product training-videos.

Emergency Deploys

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

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

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

When an emergency deploy is executed, an alert is posted to the #incident-command Slack channel for visibility to the on-call team. A post-mortem remediation issue is auto-created in the github/forced-deploys 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 deploy, which is then reviewed by their manager and an site reliability engineering (SRE) team member.

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-deploy steps to help ensure no further issues require resolution.

Complementary User Entity Controls (CUECs)

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

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

Criteria	Complementary User Entity Controls
CC6.1	• Customer administrators are responsible for setting their authentication policy for GitHub Enterprise Cloud user accounts. GitHub recommends the use of SAML/Single Sign On with the customer's identity provider.
CC5.3	• Customer administrators are responsible for establishing policy and guidelines for the use of GitHub Copilot Business by customer developers.
CC5.3	• Customer administrators are responsible for configuring GitHub Copilot Business to block suggestions matching public code
CC6.4 CC6.5 CC7.2	• Customers are responsible for deploying physical security and environmental controls for all devices and access points residing at their operational facilities, including remote employees or at-home agents for which the customer allows connectivity.

Subservice Organization and Complementary Subservice Organization Controls (CSOCs)

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

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

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

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

Criteria	Complementary Subservice Organization Controls
CC6.1	• Microsoft Azure encrypts customer data at rest. • Microsoft Azure corrects functioning of tenant isolation of its cloud services. • Microsoft Azure OpenAI maintains the functioning and performance of the AI models configured by GitHub Copilot Business.
CC6.4	• Microsoft Azure, CoreSite, Sabey, Equinix, and QTS restrict data center access to authorized personnel. • Microsoft Azure, CoreSite, Sabey, Equinix, and QTS monitor data centers 24/7 by closed circuit cameras and security personnel.
CC6.5 CC6.7	• Microsoft Azure securely decommissions and physically destroys production assets in its control.

Criteria	Complementary Subservice Organization Controls
CC7.2	• Microsoft Azure, CoreSite, Sabey, Equinix, and QTS install fire suppression and detection and environmental monitoring systems at the data centers. • Microsoft Azure, CoreSite, Sabey, Equinix, and QTS protect data centers against a disruption in power supply to the processing environment by an uninterruptible power supply (UPS). • Microsoft Azure, CoreSite, Sabey, Equinix, and QTS oversee the regular maintenance of environmental protections at data centers.

Specific Criteria Not Relevant to the System

There were no specific security Trust Services Criteria as set forth in TSP Section 100, *2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy (With Revised Points of Focus—2022)* (2017 TSC) that were not relevant to the system as presented in this report.

Report Use

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

Section 4

Trust Services Criteria and Related Controls Relevant to the Security Category

Trust Services Criteria and Related Controls Relevant to the Security Category

Control Environment	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
CC1.1	The entity demonstrates a commitment to integrity and ethical values.
	Individuals with access to the in-scope environment acknowledge and sign off on the Handbook for Hubbers as part of the onboarding process. The Handbook includes the GitHub standards of conduct, security policy awareness and responsibilities, and other information security topics.
	Employees sign a confidentiality agreement upon hire. This agreement prohibits the disclosure of information and other data to which the employee has been granted access during employment and after termination.
	GitHub management evaluates and provides feedback to direct reports annually.
	The Company has documented disciplinary actions within the Handbook for Hubbers for employees and contractors who violate the code of conduct.
	New employees undergo background screening as permissible by local laws and regulations.
CC1.2	The board of directors demonstrates independence from management and exercises oversight of the development and performance of internal control.
	The Senior Leadership team sets Company-wide objectives and reviews key results semiannually. Operational objectives and key results defined across the in-scope domains tie back to meeting those Company-wide organizational objectives, including security and risk objectives. The Senior Leadership team reports and reviews progress towards meeting those objectives semiannually.
	The board of directors contains members that are independent of the Company and has documented oversight responsibilities relative to internal control.
CC1.3	Management establishes, with board oversight, structures, reporting lines, and appropriate authorities and responsibilities in the pursuit of objectives.
	People Operations maintains a current organizational chart that outlines Hubbers' roles and is made available to employees and contractors.

Control Environment	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
	The board of directors contains members that are independent of the Company and has documented oversight responsibilities relative to internal control.
	The GitHub Information Security and Privacy Management System (ISPMS) defines information security practices, including information system management, access controls, software development, and the definition of roles and responsibilities. This policy is published to internal personnel via the GitHub intranet. The Chief Security Officer (CSO) reviews and approves the policy annually.
	Hubber responsibilities are communicated through documented job descriptions, which are stored internally.
CC1.4	The entity demonstrates a commitment to attract, develop, and retain competent individuals in alignment with objectives.
	Employees complete security and privacy awareness training upon hire and annually thereafter.
	Hubber responsibilities are communicated through documented job descriptions, which are stored internally.
	GitHub management evaluates and provides feedback to direct reports annually.
CC1.5	The entity holds individuals accountable for their internal control responsibilities in the pursuit of objectives.
	GitHub management evaluates and provides feedback to direct reports annually.
	Individuals with access to the in-scope environment acknowledge and sign off on the Handbook for Hubbers as part of the onboarding process. The Handbook includes the GitHub standards of conduct, security policy awareness and responsibilities, and other information security topics.
	The Company has documented disciplinary actions within the Handbook for Hubbers for employees and contractors who violate the code of conduct.
	Hubber responsibilities are communicated through documented job descriptions, which are stored internally.

Information and Communication	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
CC2.1	The entity obtains or generates and uses relevant, quality information to support the functioning of internal control.
	The Security Governance, Risk, Compliance & Communication (GRCC) team assesses internal security controls annually through sample-based testing to help ensure that control design and operation continue to meet defined criteria and system requirements.
	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.
	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.
	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.
	The security owner subscribes to industry security bulletins and email alerts and uses them to monitor the impact of emerging technologies and security on the production systems.
CC2.2	The entity internally communicates information, including objectives and responsibilities for internal control, necessary to support the functioning of internal control.
	Employees complete security and privacy awareness training upon hire and annually thereafter.
	GitHub management evaluates and provides feedback to direct reports annually.
	Hubber responsibilities are communicated through documented job descriptions, which are stored internally.
	Technical reviews are performed for new features and major changes to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production.
	The GitHub Handbook for Hubbers outlines the process for anonymously reporting potential security issues or fraud concerns.

Information and Communication

TSC Reference	Trust Services Criteria and Applicable Control Activities
CC2.3	The entity communicates with external parties regarding matters affecting the functioning of internal control.
	Customer commitments and responsibilities are communicated through GitHub's Data Protection Agreement and customer-provided contracts.
	Formal information sharing agreements are in place with critical vendors and subservice organizations. These agreements include confidentiality commitments applicable to that entity.
	GitHub posts notifications of new releases and customer-impacting changes to the GitHub blog.
	The GitHub external website communicates to internal and external users the description of the in-scope systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints.
	An external-facing status webpage is used to document and communicate current information on service availability to internal and external users.

Risk Assessment

TSC Reference	Trust Services Criteria and Applicable Control Activities
CC3.1	The entity specifies objectives with sufficient clarity to enable the identification and assessment of risks relating to objectives.
	The Company specifies its objectives in its annual risk assessment to enable the identification and assessment of risk related to the objectives.
	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.
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.
	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.
	Annually, GitHub performs a risk assessment that identifies, tracks, and monitors changes to in-scope systems security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. The risk assessment considers the potential for fraud and how fraud may impact the achievement of objectives.
CC3.3	The entity considers the potential for fraud in assessing risks to the achievement of objectives.
	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.
	Annually, GitHub performs a risk assessment that identifies, tracks, and monitors changes to in-scope systems security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. The risk assessment considers the potential for fraud and how fraud may impact the achievement of objectives.

Risk Assessment

TSC Reference	Trust Services Criteria and Applicable Control Activities
CC3.4	The entity identifies and assesses changes that could significantly impact the system of internal control.
	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.
	Annually, GitHub performs a risk assessment that identifies, tracks, and monitors changes to in-scope systems security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. The risk assessment considers the potential for fraud and how fraud may impact the achievement of objectives.
	A configuration management tool (CMT) maintains the state of systems to an approved baseline and reverts unapproved changes, when they are detected, back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review.
	A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution.
	The Company maintains a Bug Bounty Program that offers rewards for discovered vulnerabilities and is designed to ensure the security and integrity of the platform through treating these findings.

Monitoring Activities	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
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.
	The Security Governance, Risk, Compliance & Communication (GRCC) team assesses internal security controls annually through sample-based testing to help ensure that control design and operation continue to meet defined criteria and system requirements.
	A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution.
	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.
	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.
	Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repo. Annually, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports.
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.
	The Security Governance, Risk, Compliance & Communication (GRCC) team assesses internal security controls annually through sample-based testing to help ensure that control design and operation continue to meet defined criteria and system requirements.
	Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repo. Annually, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports.

Control Activities

TSC Reference	Trust Services Criteria and Applicable Control Activities
CC5.1	The entity selects and develops control activities that contribute to the mitigation of risks to the achievement of objectives to acceptable levels.
	As part of its annual risk assessment, management selects and develops manual and IT general control activities that contribute to the mitigation of identified risks.
	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.
CC5.2	The entity also selects and develops general control activities over technology to support the achievement of objectives.
	As part of its annual risk assessment, management selects and develops manual and IT general control activities that contribute to the mitigation of identified risks.
	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.
CC5.3	The entity deploys control activities through policies that establish what is expected and in procedures that put policies into action.
	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and to address breach or privacy considerations, which varies depending on the severity of the incident.
	Formal procedures are documented that outline the process the Company's staff follows to perform the following system access control functions: - Authorizing and granting new user access - Maintaining existing user access - Managing entitlements - Revoking existing user access - Restricting access based on entitlement membership and access request approval
	GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes.

Control Activities

TSC Reference	Trust Services Criteria and Applicable Control Activities
	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.
	Formal procedures that outline requirements for vulnerability management are documented and include the following components: - Methods for identifying vulnerabilities and frequency - Assessing the severity of identified vulnerabilities - Prioritizing and implementing remediation or mitigation activities for identified vulnerabilities based on severity and defined timelines - Handling of system components for which no measures are initiated to remediate or mitigate vulnerabilities
	Formal policies and procedures that outline the requirements for vendor management are documented and include the following components: - Maintaining a list of critical vendors - Requirements for the assessment of risks resulting from the procurement of third-party services - Requirements for the classification of third parties - Information security requirements for the processing, storage, or transmission of information by third parties - Requirements for dealing with vulnerabilities, security incidents, and malfunctions - Specifications for the contractual agreement and monitoring of third-party vendor requirements - Requirements for critical vendors to maintain their own security practices and procedures - Annually reviewing attestation reports for Tier 1 vendors or performing a vendor risk assessment
	Formal policies and procedures that outline the technical and organizational safeguards for change management of system components are documented and include the following components: - Change management roles and responsibilities - Criteria for risk assessment, categorization, and prioritization of changes - Approvals for implementation of changes - Requirements for the performance and documentation of tests, including rollback plans - Requirements for segregation of duties during development, testing, and release of changes - Requirements for the implementation and documentation of emergency changes
	A formal security and software development life cycle (SDLC) methodology is in place that governs the project planning, design, acquisition, testing, implementation, maintenance, and decommissioning of information systems and related technologies.

Control Activities	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
	GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures.
	Formal data retention and disposal procedures are documented to guide the secure retention and disposal of Company and customer data.
	Policies and procedures derived from the information security policy are documented, version controlled, reviewed at least annually, approved by management, and communicated to authorized users.

Logical and Physical Access Controls

TSC Reference	Trust Services Criteria and Applicable Control Activities
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.
	Authentication to the following in-scope production system components requires unique usernames and passwords or authorized Secure Shell (SSH) keys: - Network - Operating system (OS) - Application(s) - Data stores - Azure console - Firewalls - Log data - Backup data
	A unique account and password are required to authenticate customers to their organization for in-scope systems.
	Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token.
	Remote access to in-scope production systems is restricted through Virtual Private Networks (VPN), bastion hosts, or Security Assertion Markup Language (SAML) enforcing Hypertext Transfer Protocol Secure (HTTPS) reverse proxy, which require multi-factor authentication (MFA).
	VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.
	Passwords for in-scope system components are configured according to the Company's policy, which required the following (unless there is a system limitation): - 8 characters long, if it includes a number and a lowercase letter or - 15 characters long with any combination of characters
	Test environments are logically separated from production environments.

Logical and Physical Access Controls	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
	GitHub tracks production components and third-party services to help ensure that various aspects of the system inventory are reviewed and updated quarterly.
	Customer Git repositories are encrypted at rest.
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.
	GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role and require manager approval prior to being provisioned. GitHub's internal systems are configured to automatically provision non-role-based entitlements only after the user's manager reviews and approves the access request.
	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.
	Semi-annually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm that users with this access are restricted based on the principle of least privilege.
	The Infrastructure team reviews physical access to production data centers annually; any unapproved accounts are removed.
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.
	Privileged access to the following in-scope production system components is restricted to authorized users with a business need: - Network - OS - Application(s) - Data stores - Azure console - Firewalls - Log data - Backup data

Logical and Physical Access Controls	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
	GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role and require manager approval prior to being provisioned. GitHub's internal systems are configured to automatically provision non-role-based entitlements only after the user's manager reviews and approves the access request.
	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.
	Semi-annually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm that users with this access are restricted based on the principle of least privilege.
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.
	The Company's production environment is hosted at third-party data centers, which are carved out for the purposes of this report.
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.
	Data center media used to store production data are destroyed onsite. Certificates verifying media destruction are documented and retained.
	GitHub tracks production components and third-party services to help ensure that various aspects of the system inventory are reviewed and updated quarterly.
	Formal data retention and disposal procedures are documented to guide the secure retention and disposal of Company and customer data.
CC6.6	The entity implements logical access security measures to protect against threats from sources outside its system boundaries.
	Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token.
	Remote access to in-scope production systems is restricted through Virtual Private Networks (VPN), bastion hosts, or Security Assertion Markup Language (SAML) enforcing Hypertext Transfer Protocol Secure (HTTPS) reverse proxy, which require multi-factor authentication (MFA).

Logical and Physical Access Controls

TSC Reference	Trust Services Criteria and Applicable Control Activities
	VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.
	GitHub uses SSH to authenticate to back-end production resources through a bastion host.
	Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary.
	Firewall rulesets are reviewed at least annually. Change tickets are created to track any firewall modifications as a result of the review.
	Production traffic is encrypted when transmitted over the public internet.
	An intrusion detection system (IDS) is used to provide continuous monitoring of the Company's network and early detection of potential security breaches. Alerts are configured to notify administrators to investigate and take appropriate action based on the severity of the alert.
	Infrastructure supporting the service is patched as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.
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.
	Production traffic is encrypted when transmitted over the public internet.
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.
	Threat detection software is installed on Company endpoints to monitor for malicious software or unauthorized activity.
	An intrusion detection system (IDS) is used to provide continuous monitoring of the Company's network and early detection of potential security breaches. Alerts are configured to notify administrators to investigate and take appropriate action based on the severity of the alert.
	Infrastructure supporting the service is patched as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.

System Operations

TSC Reference	Trust Services Criteria and Applicable Control Activities
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.
	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.
	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.
	Annually, GitHub performs a risk assessment that identifies, tracks, and monitors changes to in-scope systems security, related compliance risk factors, and fraud. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution. The risk assessment considers the potential for fraud and how fraud may impact the achievement of objectives.
	A configuration management tool (CMT) maintains the state of systems to an approved baseline and reverts unapproved changes, when they are detected, back to the required configuration. Changes to the approved configuration baseline are completed through pull requests requiring peer review.
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.
	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.
	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.
	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.

System Operations	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
	An intrusion detection system (IDS) is used to provide continuous monitoring of the Company's network and early detection of potential security breaches. Alerts are configured to notify administrators to investigate and take appropriate action based on the severity of the alert.
	An infrastructure monitoring tool is utilized to monitor system or infrastructure availability and performance and generates alerts when specific, predefined thresholds are met.
	A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution.
	Infrastructure supporting the service is patched as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.
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.
	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and to address breach or privacy considerations, which varies depending on the severity of the incident.
	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.
	A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution.
	The Security Operations team scans internal and external systems monthly, reviews identified vulnerabilities, and shares confirmed threats with responsible stakeholders.
	System owners remediate vulnerabilities in accordance with established service-level agreements (SLAs) based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.

System Operations

TSC Reference	Trust Services Criteria and Applicable Control Activities
	Infrastructure supporting the service is patched as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.
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.
	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and to address breach or privacy considerations, which varies depending on the severity of the incident.
	All incidents related to security and involving a data breach are logged, tracked, evaluated, and communicated to affected parties by management until the Company has recovered from the incidents.
CC7.5	The entity identifies, develops, and implements activities to recover from identified security incidents.
	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and to address breach or privacy considerations, which varies depending on the severity of the incident.
	All incidents related to security and involving a data breach are logged, tracked, evaluated, and communicated to affected parties by management until the Company has recovered from the incidents.
	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.

Change Management	
TSC Reference	**Trust Services Criteria and Applicable Control Activities**
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.
	An independent reviewer evaluates and approves software and infrastructure change requests via a GitHub pull request.
	Technical reviews are performed for new features and major changes to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production.
	GitHub engineering code owners create and execute automated regression test cases to address functionality and security requirements.
	Application changes pass automated continuous integration testing prior to deployment to the production environment.
	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.
	Test environments are logically separated from production environments.
	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.
	The Production Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems.
	Infrastructure supporting the service is patched as a part of routine maintenance and as a result of identified vulnerabilities to help ensure that servers supporting the service are hardened against security threats.

Risk Mitigation

TSC Reference	Trust Services Criteria and Applicable Control Activities
CC9.1	The entity identifies, selects, and develops risk mitigation activities for risks arising from potential business disruptions.
	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.
	Security incident response and escalation policies and procedures are documented and provide guidance to Company personnel for detecting, responding to, and recovering from security events and incidents and to address breach or privacy considerations, which varies depending on the severity of the incident.
	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.
CC9.2	The entity assesses and manages risks associated with vendors and business partners.
	Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repo. Annually, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports.
	Formal information sharing agreements are in place with critical vendors and subservice organizations. These agreements include confidentiality commitments applicable to that entity.