GitHub

REPORT ON GITHUB'S ENTERPRISE CLOUD RELEVANT TO SECURITY (SOC 3 REPORT)

FOR THE PERIOD OCTOBER 1, 2019 TO SEPTEMBER 30, 2020

THE cadence GROUP

THE cadence GROUP

O 801.349.1360
F 866.326.6612

PO Box 711190
Salt Lake City, Utah 84171
www.thecadencegroup.com

Section I – Report of Independent Service Auditors

To: GitHub, Inc.

Scope

We have examined GitHub's accompanying assertion, titled "GitHub's Assertion" (assertion), that the controls within GitHub's Enterprise Cloud were effective throughout the period October 1, 2019 to September 30, 2020, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the trust services criteria relevant to security, availability, and confidentiality (applicable trust services criteria) set forth in TSP section 100, *2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy* (AICPA, *Trust Services Criteria*).

Service Organization's Responsibilities

GitHub is responsible for its service commitments and system requirements and for designing, implementing, and operating effective controls within the system to provide reasonable assurance that GitHub's service commitments and system requirements were achieved. GitHub has provided the accompanying assertion about the effectiveness of controls within the system. When preparing its assertion, GitHub is responsible for selecting, and identifying in its assertion, the applicable trust services criteria, and for having a reasonable basis for its assertion by performing an assessment of the controls within the system.

Service Auditor's Responsibilities

Our responsibility is to express an opinion, based on our examination, on whether management's assertion that controls within the system were effective throughout the period to provide reasonable assurance that the service organization's service commitments and system requirements were achieved based on the applicable trust services criteria. Our examination was conducted in accordance with attestation standards established by the American Institute of Certified Public Accountants. Those standards require that we plan and perform our examination to obtain reasonable assurance about whether management's assertion is fairly stated, in all material respects. We believe that the evidence we obtained is sufficient and appropriate to provide a reasonable basis for our opinion.

Our examination included:

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

THE cadence GROUP

O 801.349.1360
F 866.326.6612

PO Box 711190
Salt Lake City, Utah 84171
www.thecadencegroup.com

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

Service Auditor's Independence and Quality Control

We have complied with the independence and other ethical requirements of the Code of Professional Conduct established by the AICPA. We applied the Statements on Quality Control Standards established by the AICPA and, accordingly, maintain a comprehensive system of quality control.

Inherent Limitations

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

Opinion

In our opinion, management's assertion that the controls within GitHub's system were effective throughout the period October 1, 2019 to September 30, 2020, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria, is fairly stated, in all material respects.

Cadence Assurance LLC

November 13, 2020
Salt Lake City, Utah

GitHub

Section II – GitHub's Assertion

We, are responsible for designing, implementing, operating, and maintaining effective controls within GitHub's Enterprise Cloud throughout the period October 1, 2019 to September 30, 2020, to provide reasonable assurance that GitHub's service commitments and system requirements relevant to security, availability, and confidentiality were achieved. Our description of the boundaries of the system is presented in Attachment A and identifies the aspects of the system covered by our assertion.

We have performed an evaluation of the effectiveness of the controls within the system throughout the period October 1, 2019 to September 30, 2020, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the trust services criteria relevant to security, availability, and confidentiality (applicable trust services criteria) set forth in TSP section 100, *2017 Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy* (AICPA, *Trust Services Criteria*). GitHub's objectives for the system, in applying the applicable trust services criteria, are embodied in its service commitments and system requirements relevant to the applicable trust services criteria. The principal service commitments and system requirements related to the applicable trust services criteria are presented in Attachment B.

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

We assert that the controls within the system were effective throughout the period October 1, 2019 to September 30, 2020, to provide reasonable assurance that GitHub's service commitments and system requirements were achieved based on the applicable trust services criteria.

GitHub, Inc.
November 13, 2020

GitHub

Attachment A – GitHub's Description of the Boundaries of Its Enterprise Cloud

Company Overview

GitHub, an independently operated Microsoft subsidiary, generated its first commit in 2007. It's headquartered in San Francisco, California, with additional offices in Boulder, Colorado; Tokyo, Japan; and Amsterdam, Netherlands. GitHub currently employs around 1,000 employees, with approximately 65 percent of the workforce remote.

System Description

GitHub is a web-based software development platform built on the Git version control software. Primarily used for software code, GitHub offers the distributed version control and source code management functionality of Git with additional features and enhancements. Specifically, it provides access control and several collaboration features including bug tracking, feature requests, task management, and wikis.

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

Organizations

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

Code Hosting

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

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

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

Code Management

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

GitHub

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

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

Protected branches allow for better quality code management. Repositories can be configured to require status checks, such as continuous integration tests, reducing both human error and administrative overhead.

Project Management

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

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

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

Team Management

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

GitHub Teams organizes people, provides level-up access with administrative roles, and tunes permissions for nested teams. Discussion threads keep conversations on topic using moderation tools, like issue and pull request locking, to help teams stay focused on code. For maintaining open source projects, user blocking reduces noise and keeps conversations productive.

Documentation

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

System Boundaries

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

Subservice Organizations

GitHub uses multiple subservice organizations in conjunction with providing its Enterprise Cloud product. GitHub uses Sabey, QTS, Coresite, and Equinix to provide colocation data center services, and Amazon Web Services (AWS) to provide infrastructure hosting. These subservice organizations are excluded from the scope of this report. The expected controls for which they are responsible are included in Attachment D, titled *Complementary Subservice Organization Controls*.

System Components

The components of the Enterprise Cloud product include infrastructure, software, people, procedures, and data. These components include the applications and services housed within GitHub's colocation data centers or internally managed systems and services related to maintaining the Enterprise Cloud product.

Infrastructure

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

- Border/edge routers, GitHub load balancers, application layer proxies, and firewalls are the systems that connect to the internet. These routers, application proxies, and firewalls are the first line of defense in protecting the system.
- Web front-end servers are the forward-facing HTTPS servers for Enterprise Cloud. These servers provide the feature set for Enterprise Cloud.
- Application servers are used for processing asynchronous jobs or back-end processes required to support the web front-end. This processing might include replication between physical data centers, sending webhooks to repository integrations, sending emails, or performing other back-end processing.
- Email servers send email notifications to users or receive email issue comments from users.
- Git proxy servers manage the Git interaction between the users and the GitHub file servers.
- API front-end servers are the interface to any client using the Advanced Programming Interface (API) to interact with GitHub programmatically.
- VPN servers are used by GitHub employees to create a secure channel and an initial layer of authorization for employees who are developing or maintaining Enterprise Cloud.
- Bastion hosts, also referred to as jump hosts, are used by GitHub employees to manage the Enterprise Cloud environment.
- Database servers store the issues, milestones, and other project management information.
- Git Infrastructure File Servers are where the code is stored for Git code repositories.

Data Centers

Enterprise Cloud infrastructure is hosted in geographically distributed, third-party data centers, specifically located in Virginia (Dulles and Ashburn) and Washington state (Seattle and Tukwila). Full production Git data is replicated between the Northern Virginia and Washington data centers to help ensure rapid recovery of the service in the event of failure or data loss.

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

Software

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

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

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

People

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

- Senior Leadership
- Security
- Product & Application Engineering
- IT
- Infrastructure
- People Operations
- Legal
- GitHub Support

Procedures

GitHub maintains programmatic (automated) and manual procedures involved in the operation of the Enterprise Cloud product. These procedures are developed and documented within the GitHub repositories maintained by every team to provide end-user documentation and guidance on the multitude of operational functions performed daily by GitHub Security and Product engineers, developers, administrators, and support. These procedures are drafted in alignment with the overall Information Security policies and are updated and approved as necessary for changes in the business, at a minimum annually.

GitHub policies establish procedures and controls to enable security, efficiency, availability, and quality of service. The GitHub Entity Security Policy and related policy statements define information security practices, roles, and responsibilities. The Entity Security Policy outlines the security roles and

responsibilities for the organization and expectations for employees, contractors, and third parties utilizing GitHub systems or data.

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

Data

GitHub uses repository data to connect users to relevant tools, people, projects, and information. Repositories are categorized as either public or private. Public repositories can be viewed by anyone, including people who are not GitHub users. Private repositories are only visible to the repository owner and collaborators that the owner specified. GitHub aggregates metadata and parses content patterns to deliver generalized insights within the product.

If a private repository is opted in for data use to take advantage of any of the capabilities of the security and analysis features, then GitHub performs a read-only analysis of that specific private repository's Git contents. If a private repository is not opted in for data use, its private data, source code, or trade secrets are classified internally as restricted, and they are maintained as confidential and private consistent with GitHub's Terms of Service. Private data exchanged with GitHub is transmitted over SSL. Send and receipt of private data is done over SSH authenticated with keys, or over HTTPS, using a GitHub username and password.

For more information about GitHub's use of data, refer to https://docs.github.com/en/free-pro-team@latest/github/understanding-how-github-uses-and-protects-your-data/about-githubs-use-of-your-data.

Internal Control Framework

GitHub has adopted an internal control framework to meet its security commitments. This framework includes the following aspects: control environment, risk assessment, control activities, information and communication, and monitoring.

Additionally, 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. See Attachment C for identified complementary user entity controls.

Control Environment

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

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

Risk Assessment

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.

Risk Identification

GitHub has defined risk management processes to identify and manage risks that may affect the system's security. At least annually, Security leadership performs a risk assessment to identify and evaluate potential threats to the effectiveness of the control environment.

Risk Mitigation

A risk register is created as part of the annual risk assessment. The Security GRC team maintains the risk register with updates from the broader Security team to help ensure security and technology-related compliance risks are identified, tracked, and mitigated.

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 GRC 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 are not moved forward for procurement.

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

Risk Reporting

Summary reporting on security risk areas is included in annual leadership reporting as part of the annual security risk assessment. Leadership, including Security leadership and the Senior Vice President of Technology, reviews and approves this annual risk report.

Control Activities

Controls have been implemented to address system and data risks. Controls have been designed and implemented in the following areas:

- System Inventory
- User Authentication
- Network Security
- Encryption
- User Provisioning
- Antivirus
- Configuration Management
- Vulnerability Management
- Bug Bounty
- Penetration Testing
- System Monitoring
- Incident Response
- Change Management

Information and Communication

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

Internal Communications

GitHub has published policies and procedures, both included in the Hubber Handbook and published separately, outlining the responsibility of employees to report security and operational failures, incidents, system problems, and complaints. The document owner(s) and Security leadership review

and approve security policies and procedures annually. Significant changes to policies result in communication to personnel regarding policy updates.

Every Hubber, depending on their role, is responsible for designing and executing work and services in a secure manner in alignment with company standards and policies, and for reporting issues and findings that impact security up their management chain or directly to the Security team.

External Communications

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

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

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

The user community can communicate directly with GitHub. GitHub Support receives reports of security issues and many other end-user concerns and questions via email or through the 'Contact Us' web form. There is a defined process using the customer-facing ticketing system, Halp, for managing, investigating, and resolving these types of issues in a timely manner.

Monitoring

The Security-GRC 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-GRC team conducts internal control assessments annually to assess the effectiveness of the internal control environment. Control assessment results are documented in internal GitHub repositories. Issues are identified and escalated to the relevant internal teams to resolve control design or effectiveness issues, and the status of the operating effectiveness of controls, identified gaps, and remediation efforts are reported up to the Senior Leadership team for awareness on a quarterly basis.

GitHub

Attachment B – Principal Service Commitments and System Requirements

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

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

Attachment C – Complementary User Entity Controls

GitHub's control environment was designed under the assumption that certain controls would be implemented by user organizations, the application of which is necessary to meet certain trust services criteria identified in this report. This section highlights those internal control responsibilities GitHub believes should be present at each customer and has considered in developing its controls reported herein. GitHub customers should evaluate their own control environment to assess if the following controls are implemented and operating effectively. These complementary user entity controls do not represent a comprehensive list of controls that should be employed by GitHub customers, but are rather a summary of controls necessary to meet the stated trust services criteria presented in this report. These controls include the following:

- Customers are responsible for enabling SAML for their Enterprise Cloud accounts. (CC6.1)
- Customers are responsible for enabling two-factor authentication and ensuring members and collaborators require two-factor authentication. (CC6.1, CC6.6)
- Customers are responsible for creating and managing their Organization and Teams, including the proper configuration of access permissions to repositories. (CC6.2, CC6.3)
- Customers are responsible for inviting, removing, and managing users in their Organization and Teams, including granting of permission levels and access to repositories, and periodic review of Organization users and outside collaborators. (CC6.2, CC6.3)
- Customers are responsible for ensuring authorized users are appointed as Organization owners for administration of the Organization. (CC6.2, CC6.3)
- Customers are responsible for maintaining an effective onboarding and offboarding process for their own employees and contractors. (CC6.2, CC6.3)
- Customers are responsible for reviewing events in the security logs. (CC7.1)
- Customers are responsible for administering and configuring repositories, including permissions, enabling required reviews for pull requests, and enabling required status checks before merging. (CC8.1)
- Customers are responsible for reviewing and authorizing third-party applications, properly configuring the GitHub Application Programming Interface (API), including connecting with third-party applications, and managing third-party application access to their own repositories and data, where applicable. (CC9.2)

GitHub

Attachment D – Complementary Subservice Organization Controls

GitHub contracts with Amazon Web Services (AWS), Coresite, Sabey, QTS, and Equinix to provide data center hosting and infrastructure support. Controls managed by these third-party subservice providers are not included in the scope of this report. Expected subservice provider controls that have an effect on specific criteria are included below.

Criteria	Expected Controls	Relevant Subservice Providers
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.	Access to the in-scope systems requires users to authenticate using a valid, unique user ID and password before being granted access. User content is segregated and made viewable only to authorized individuals.	AWS
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.	New user accounts are approved by appropriate individuals prior to being provisioned. User accounts are removed when access is no longer needed. User accounts are periodically reviewed to verify the accounts, and their permissions, are current and appropriate.	AWS

Criteria	Expected Controls	Relevant Subservice Providers
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.	Access modifications are approved by appropriate individuals prior to being provisioned. User accounts are removed when access is no longer needed. User accounts are periodically reviewed to verify the accounts, and their permissions, are current and appropriate.	AWS
CC6.4 – The entity restricts physical access to facilities and protected information assets (for example, data center facilities, back-up media storage, and other sensitive locations) to authorized personnel to meet the entity's objectives.	Only authorized users have access to the physical facilities securing the system.	AWS QTS Sabey Coresite Equinix
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.	Production media is securely decommissioned and physically destroyed prior to being removed from the data center.	AWS
CC6.6 – The entity implements logical access security measures to protect against threats from sources outside its system boundaries.	Network security mechanisms restrict external access to the production environment.	AWS

Criteria	Expected Controls	Relevant Subservice Providers
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.	Access to customer data is restricted to appropriate users. Customer data is protected during transmission.	AWS QTS Sabey Coresite Equinix
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.	Antivirus or antimalware solutions are installed to detect or prevent unauthorized or malicious software.	AWS
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.	Vulnerabilities are logged and tracked to resolution.	AWS
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.	Security events on system components are monitored and evaluated to determine potential impact per policy.	AWS QTS Sabey Coresite Equinix

Criteria	Expected Controls	Relevant Subservice Providers
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 events are reviewed to determine whether they should be elevated to an incident.	AWS QTS Sabey Coresite Equinix
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 events deemed incidents are remediated and communicated.	AWS QTS Sabey Coresite Equinix
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.	System changes are documented, tested, and approved prior to migration to production.	AWS QTS Sabey Coresite Equinix