GitHub

Study Guide
GitHub Advanced Security

Get exam-ready for the GitHub Advanced Security Certification with our comprehensive study guide. We've curated the essential resources and learning activities to better prepare you for the exam and boost your chances of success.

Audience Profile

This exam is designed for experienced professionals in the field of software development and security. This certification is designed for individuals who have a deep understanding of GitHub and its security features, as well as hands-on experience in securing software development workflows.

Objective Domains

An objective domain for a certification exam, often referred to as a "domain" or "exam domain," is a structured outline or framework that defines the specific knowledge, skills, and topics that the certification exam will cover. It provides a clear roadmap for what candidates should expect to encounter on the exam and what they need to study and prepare for.

The domains provided in this study guide are intended to provide insight into the topic categories covered in the GitHub Advanced Security exam, along with the learning objective within each domain.

Domain Breakdown	Exam Percentages
Domain 1: Describe the GHAS security features and functionality	10%
Domain 2: Configure and use secret scanning	10%
Domain 3: Configure and use dependency management	15%
Domain 4: Configure and use code scanning	15%
Domain 5: Use code scanning with CodeQL	20%
Domain 6: Describe GitHub Advanced Security best practices	20%
Domain 7: Configure GitHub Advanced Security tools in GitHub Enterprise	10%

Recommendations and Best Practices for Success

To increase your chances of success in passing the GitHub Advanced Security exam, candidates should have a deep understanding of GitHub and its security features, as well as hands-on experience in securing software development workflows. The recommended learning paths for this exam provide you with an in-depth study of the learning content, followed by hands-on exercises and preparation assessment questions that were created to enable you to fine-tune your knowledge and readiness for the certification exam.

Content Resources

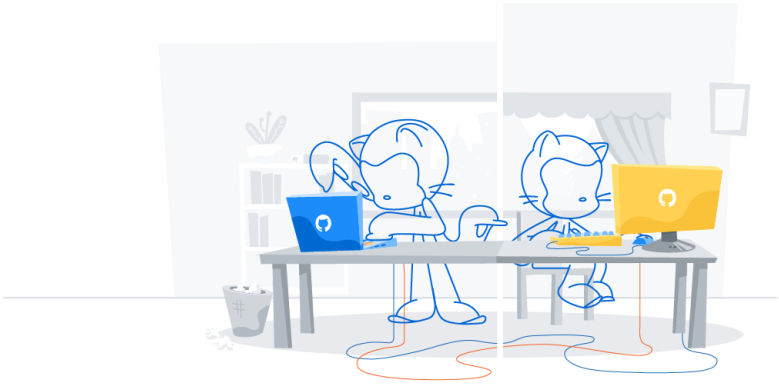
The following resources have been created in collaboration with GitHub as recommended content that covers the learning objectives in each domain for the GitHub Advanced Security exam. Both Microsoft Learn and LinkedIn Learning provide a complete learning path for the exam, but offer a different learning experience.

Microsoft Learn

The GitHub Advanced Security learning path on MS Learn provides a robust collection of learning modules designed to prepare you for the GitHub Advanced Security exam. Learn how to secure your code with advanced security features at every stage of your development lifecycle. GitHub Advanced Security is an add-on to GitHub Enterprise that allows you to use security features, such as secret scanning, code scanning, and dependency management on your private repositories. The following modules will walk you through GitHub's advanced security features and provide you with the skills needed to recognize, apply, and evaluate these features within your own GitHub environment.

LinkedIn Learning

Discover the art of safeguarding your code using advanced security techniques throughout the entire software development process by going through the Prepare for the GitHub Advanced Security Certification learning path on LinkedIn Learning. With GitHub Advanced Security, you gain access to a suite of robust security tools, including secret scanning, code scanning, and dependency management, tailor-made for your private repositories. This video-based learning path comprises a series of engaging courses that will guide you through the intricacies of GitHub's advanced security capabilities. By the end of this learning path, you'll be well-equipped with the knowledge and expertise to seamlessly apply, assess, and maximize these security features within your GitHub environment.



Domain 1: Describe the GHAS security features and functionality

Contrast GHAS features and their role in the security ecosystem
Differentiate the security features that come automatically for open source projects, and what features are available when GHAS is paired with GHEC or GHES
Describe the features and benefits of Security Overview
Describe the differences between secret scanning and code scanning
Describe how secret scanning, code scanning, and Dependabot create a more secure software development life cycle
Contrast a security scenario with isolated security review and an advanced scenario, with security integrated into each step of the software development life cycle

Explain and use specific GHAS features
Describe how vulnerable dependencies are identified (by looking at the manifest files and comparing with databases of known vulnerabilities)
Explain how to act on alerts from GHAS
Explain the implications of ignoring an alert
Explain the role of a developer when they discover a security alert
Describe the differences in access management to view alerts for different security features
Describe a security policy in a GitHub repository
Identify where to use Dependabot alerts in the software development lifecycle

Domain 2: Configure and use secret scanning

Enable and use secret scanning
Describe secret scanning
Choose when secret scanning occurs
Contrast secret scanning availability for public and private repositories
Enable secret scanning for private repositories
Enable secret scanning for an organization
Explain how to pick an appropriate response to a secret scanning alert
Determine if an alert is generated for a given secret, pattern, or service provider
Determine if a given user role will see secret scanning alerts

Customize default secret scanning behavior
Configure the recipients of a secret scanning alert (also includes how to provide access to members and teams other than admins)
Describe how to exclude certain files from being scanned for secrets
Explain how to enable custom secret scanning for a repository
Explain how to enable custom secret scanning for an organization

Domain 3: Configure and use dependency management

Describe tools for managing vulnerabilities in dependencies
Define a vulnerability
Describe Dependabot alerts
Describe Dependabot security updates
Define the dependency graph
Describe how the dependency graph is generated
Describe how alerts are generated for vulnerable dependencies (driven from the dependency graph, sourced from the Github Advisory Database and from WhiteSource)

Enable and configure tools for managing vulnerable dependencies
Identify the default settings for Dependabot alerts in public and private repositories
Identify the permissions and roles required to enable Dependabot alerts
Identify the permissions and roles required to view Dependabot alerts
Enable Dependabot alerts for private repositories
Enable Dependabot alerts for organizations
Create a valid Dependabot configuration file
Configure notifications for vulnerable dependencies

Identify and remediate vulnerable dependencies
Identify a vulnerable dependency from a Dependabot alert
Identify vulnerable dependencies from a pull request
Enable Dependabot security updates
Remedy a vulnerability from a Dependabot alert in the Security tab (could include updating or removing the dependency)
Remedy a vulnerability from a Dependabot alert in the context of a pull request (could include updating or removing the dependency)
Take action on any Dependabot alerts by testing and merging pull requests

Domain 4: Configure and use code scanning

Describe and enable code scanning
Describe code scanning
List the steps for enabling code scanning in a repository using GitHub Actions (i.e. Security tab and click on “set up code scanning”, set up the GitHub Actions workflow, make any necessary modifications to the workflow)
Enable code scanning for use with a CodeQL analysis workflow
Describe how code scanning relates to GitHub Actions consumption

Use code scanning with third-party tools
Enable code scanning for use with a third-party analysis
Contrast the steps for using CodeQL versus third party analysis when enabling code scanning
Contrast how to implement CodeQL analysis in a GitHub Actions workflow versus a third-party CI tool

Configure code scanning
Describe how code scanning fits in the software development life cycle
Contrast the frequency of code scanning workflows (scheduled versus triggered by events)
Choose a triggering event for a given development pattern (for example, in a pull request and for specific files)
Edit the default template for Actions workflow to fit an active, open source, production repository

Domain 5: Use code scanning with CodeQL

Explain how CodeQL enables code scanning
Describe CodeQL
Define a QL pack, code query, code suite
Describe the default CodeQL query suites
Describe how CodeQL analyzes code and produces results, including differences between compiled and interpreted language

Use CodeQL for code scanning
Introduce a CodeQL analysis workflow to a repository
List the locations in which CodeQL queries can be specified for use with code scanning
Configure the language matrix in a CodeQL workflow
Reference a CodeQL query from a public repository within a code scanning workflow
Reference a CodeQL query from a private repository within a code scanning workflow
Reference a CodeQL query from a local directory within a code scanning workflow
Reference a configuration file within the same repository
Reference a configuration file in a remote public repository
Execute code scanning with the CodeQL command-line interface (CLI), including creating the CodeQL database, analyzing that database, and posting the SARIF results to GitHub
Contrast the steps to execute code scanning in GitHub Actions vs the CodeQL CLI

Describe how to triage code scanning results from CodeQL analysis
Describe how to view code scanning results from CodeQL analysis
Troubleshoot a failing code scanning workflow using CodeQL, including creating or changing a custom configuration in the CodeQL workflow
Follow the data flow through code using the show paths experience
Explain the reason for a code scanning alert given documentation linked from the alert
Determine if and why a code scanning alert needs to be dismissed
Describe potential shortfalls in CodeQL via model of compilation and language support
Optimize CodeQL analysis runtimes

Use third-party tools with code scanning
Explain how to upload 3rd party SARIF results via the SARIF endpoint
Explain the purpose of defining a SARIF category

Domain 6: Describe GitHub Advanced Security best practices, results, and how to take corrective measures

Describe GitHub Advanced Security best practices, results, and how to take corrective measures
Use a Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) to describe a GitHub Advanced Security alert and list potential remediation
Describe the decision-making process for closing and dismissing security alerts (documenting the dismissal, making a decision based on data)
Determine the roles and responsibilities of development and security teams on a software development workflow
Explain how to set a review cadence with security teams, when appropriate
Use security policies to instruct all contributors to better secure their repositories
Compare the code scanning alert against the repository's security policy (i.e. should we block merges with unfixed security vulnerabilities?)
Align repository branch protection configuration with written security policies

Domain 7: GitHub Advanced Security Administration

GitHub Advanced Security Administration
Explain how GitHub Advanced Security features are enabled on GitHub Enterprise Server
Explain how GitHub Advanced Security features are enabled for an organization
Set security policies for a repository
Set security policies for an organization
Describe how permissions are interpreted throughout security workflow
Locate API endpoints for GHAS features, like secret scanning, code scanning, and dependabot
List stakeholders that need to be involved in the security workflows enabled by GHAS, including their role in the workflow
Configure code scanning within a repository or organization using the default CodeQL workflow
Identify the custom build steps necessary in a CodeQL workflow