Security Guidelines for Providing and Consuming APIs

Cloud Security Alliance - Israel

cloud security alliance®

© 2021 Cloud Security Alliance – All Rights Reserved. You may download, store, display on your computer, view, print, and link to the Cloud Security Alliance at https://cloudsecurityalliance.org subject to the following: (a) the draft may be used solely for your personal, informational, non-commercial use; (b) the draft may not be modified or altered in any way; (c) the draft may not be redistributed; and (d) the trademark, copyright or other notices may not be removed. You may quote portions of the draft as permitted by the Fair Use provisions of the United States Copyright Act, provided that you attribute the portions to the Cloud Security Alliance.

Acknowledgments

Creators:

Oz Avenstein
Shahar Geiger Maor

Contributors:

Marius Aharonovich
Moshe Ferber
Reuven Harrison
Ofer Maor
Michael Roza
Eitan Satmary

Cloud Security Alliance:

Todd Edison
Frank Guanco
John Yeoh

Special Thanks:

Bowen Close

About the Cloud Security Alliance

The Cloud Security Alliance (CSA) is a not-for-profit organization with a mission to promote the use of best practices for providing security assurance within the cloud computing industry. Furthermore, it provides education on the application of cloud computing and its role in securing all other forms of computing. The CSA is led by a broad coalition of industry practitioners, corporations, associations, and other key stakeholders. For further information, visit www.cloudsecurityalliance.org and follow us on Twitter @cloudsa.

About the CSA Israel Chapter

This document was created by the Israeli chapter of the Cloud Security Alliance (CSA). The CSA Israel chapter was founded by security professionals united in a desire to promote responsible cloud adoption in the Israeli market while delivering useful knowledge and global best practices to the Israeli innovation scene. Visit our Facebook group at www.facebook.com/groups/789522244477928 for more details.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Table of Contents

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Introduction

In modern application workloads, organizations are often required to integrate their application with other parties such as Software-as-a-Service (SaaS) providers, customers applications, and business partners. Such integrations may range from granting one-time read access through ongoing static data consumption internally and all the way to exposure of APIs or application components to a third-party provider and customers.

Examples for API connectivity:

- Connecting a SaaS application – a SaaS provider connecting to an application to read or write information and provide insights
- Connecting customer platforms – connecting a customer application (CRM, ERP, BI)
- Connecting a security or monitoring provider to an internal application

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Purpose of This Document

The purpose of this document is to provide a framework for securely connecting external entities, such as customers or third parties. The document provides a usable list of security considerations in order to estimate the risk involved with specific connectivity (first part of the document) and a technical checklist for the implementation of security controls (second part of the document).

Important: Use this document if the answer to any of these questions is YES:

1. Does the new service/system require long-term integration with the company's internal systems?
2. Does the new service/system require exchange of data with a third party?
3. Will the APIs be exposed to external parties, including the public (i.e., open APIs)?

These guidelines are also highly recommended for non-public APIs (i.e., APIs are used internally or only exposed to restricted parties, such as in a B2B environment).

Document Terminology

Third Party – For the purposes of this document, third party includes the infrastructure or servers of any external entity, including customers, partners, sub-contractors, vendors, and so on.

Service Owner – The organizational entity that is responsible for the development of a certain application/API.

Platform – Usually, the team that operates the platform and gateway that enable the connectivity.

Document Scope

The document encapsulates a list of security controls for two main cases:

1. Providing APIs for an application to consume
2. Consuming APIs from other applications

Additional use cases for connectivity as direct access to infrastructure or consuming a marketplace application are described in the appendixes.

This document **will not** address the challenges of remote access to IT resources. There are other resources that address remote connectivity of individuals, including the Cloud Security Alliance Software Defined-Perimeter working group and other initiatives.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Target Audience

The target audience for this document are IT professionals who design and implement applications within an organization: software developers, software architects, security architects, IT security professionals, DevOps/DevSecOps professionals, and CISOs/Head(s) of Information Security.

Overview

The document is composed of two parts:

1. Risk assessment for connectivity – the purpose of Part 1 is to understand the risk associated with third-party connectivity and the business impact (see definition in Part 1)
2. Control checklist for secure connectivity – the purpose of Part 2 is to provide security controls for third-party access. There are two parts to this checklist: ingress access and egress access.

There are also four appendixes, covering two additional use cases of secure connectivity:

A. Mapping the controls checklist to OWASP API security top ten
B. Reading/writing internal data directly to a local infrastructure service (i.e., accessing a bucket or direct database access)
C. Connecting third-party applications or components from marketplaces
D. Mapping the document to CSA CCSK and CCAK body of knowledge

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Part 1: Risk Evaluation

In order to properly evaluate the security risks of a newly established third-party integration, there is a need to assess the risk of the integration.

As a starting point, this document offers a list of common questions that should be asked in the initial phases of a new integration project. As security risks may evolve over time, questions should be reviewed periodically so that the risk evaluation is updated. Each item on the list should be scored on a scale of 1-5 (where 5 is the highest risk).

Please note that the risk score is subject to the business context, the industry, regulations, and other external factors that should be taken into consideration by the risk evaluator. For each risk area in the below table, we specified suggested controls that can be applicable to help mitigate the risk. The full list of suggested controls can be found in the following section (Part 2: Security Checklist).

#	Risk Areas	**Low Risk (1)** Score 9-19	**High Risk (5)** Score 30-40	**Suggested Mitigation Controls**
1	**API Accessibility** (Risk depends on the type and trust of the API consumers)	Private B2B API (private API)	Any third party can consume the API (public API)	1, 2, 5, 7, 10, 11, 16, 17-26, 28, 31, 32
2	**Trust of Third Party**	Trusted source/ destination	New/untrusted	1, 7, 12, 13, 14, 15, 17, 19, 25, 28, 31
3	**Existing vs. New API** (Assuming a risk assessment was conducted to existing APIs)	Existing	New	1, 32
4	**Volume of Data Involved**	Low volume (X<10K records)	High volume (100K<X records)	7, 13, 25, 26, 28.31
5	**Sensitivity of Data**	Public, non-PII	PII/business/ financial	1, 2, 3, 5, 6, 10, 12, 14, 15, 16, 17, 18, 19, 25, 29, 30, 31
6	**Required Permissions to Conduct an Action**	Read-only for one user/service account	Wide range of permissions to all functions	1, 3, 5, 7, 14

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Risk Areas	Low Risk (1) Score 9-19	High Risk (5) Score 30-40	Suggested Mitigation Controls
7	**Integration Frequency** (continuous has inherent risks vs. one-time)	One-time	Continuous or long-term	1, 2, 3, 5, 7, 14, 18, 26, 29, 30, 31
8	**Data Retention**	X<One month	12 months<X	1, 3, 5, 7, 14, 17, 18, 23, 26, 29, 30, 31
9	**Third Party's Security and Security Compliance Status** (SOC 2, penetration test, ISO 27001, etc.)	Fully compliant	Not compliant or partially compliant	1, 3, 4, 7, 9, 10, 12, 13, 16, 17, 20, 23, 29

Business impact should be measured according to the projected span of the integration and the potential of cross-organization implementation (e.g., a one-time limited integration with no sensitive data vs. exposure of full public and robust API):

1. **Low business impact:** Small sub-system (i.e., blog, marketing landing page with low traffic)
2. **High business impact:** Cross-platform implementation that impacts revenue (i.e., exposing customer personal data)

More information about how to properly perform Cloud risk management can be found at:

- *CSA Security Guidance for Critical Areas of Cloud Computing* (Domain 2)
- *CSA Cloud Computing Auditor Knowledge* (Module 1, Unit 1.7)

Once a risk level is determined, the next phase of determining security controls begins.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Part 2: Security Checklist

API Security Guidelines

The list below consists of relevant security controls for building and publishing an API. The list is written as a "do" and "don't" checklist with a description of the task.

Note: Not all items are necessarily required for all use cases.

The guidelines are based on two common use cases:

- Section 1: A third-party reading/writing data via ingress API exposure (private or public)
- Section 2: Reading/writing data to a third-party application exposed API (egress access)

Section 1: Design for Ingress API Connectivity

The checklist below is created to assist readers in assessing the different steps required to secure API from the design phase to the ongoing monitoring phase.

#	Control	Description	Suggested Owner
		Phase 1: Design	
1	**Threat Modeling and Countermeasures**	As part of the initial design phase of an API, a threat modeling process should be performed to assess the possible threats and vulnerabilities and the associated likelihood against the attack surface. Any identified threats must be treated with appropriate countermeasure. Design should be based on best practices.	Service owner/ security
2	**Service Authentication**	Implement an authentication mechanism which is based on best practices (password policy, cross account roles), the organization's security policy, and the security risk level. For application authentication, there are a couple of secure and mature protocols for API authentication and authorization (e.g., API keys, OAuth, JWT, and more). The decision of which authentication mechanism to use depends on the scenario and the risk level.	Platform

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
3	**Least Privilege**	Authorization is granted based on the principle of "Least Privilege" – the minimum permissions and HTTP methods needed to conduct a business-required action. In order to prevent unintentional or improper uses of privilege, access from any external entity/consumer to internal resources will be limited to specific routes, based on restricted permissions.	Service owner/ platform
4	**Cross Site Request Forgery (CSRF) Protection**	In cases where the API utilizes authentication and authorization of the user or account by cookies, a CSRF protection mechanism should be employed.	Platform
5	**Authorization Decoupled from the Application**	Authorization should be decoupled from the application (e.g., implementing it as a separate service or sidecar).	Service owner/ platform
6	**Input Validation Decoupled from the Application**	Input validation should be decoupled from the application logic and specified in a format that can be reviewed (e.g., an OpenAPI Specification).	Service owner
	Phase 2: Development		
7	**Request Rate Limiting**	Implement a mechanism to detect and prevent excessive usage of the API. The legitimate usage should be defined by the service owner and according to the organizational security policy	Platform/ service owner
8	**Safe Package Usage**	While developing software, and especially API, external software packages and libraries should be checked for vulnerabilities. Patches should be applied to the latest stable version.	Service owner

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
9	**Storage of Application Secrets**	Encryption keys and application secrets should be stored in a secure location such as a vault or HSM. The development process should include a pre-commit-hook to prevent any sensitive strings such as secrets, passwords, and keys from entering the code repository. Access keys and other secrets should be rotated periodically according to the organization security policy. Delivery of API keys should be using a secure channel and encrypted in transit.	Service owner/ DevOps
10	**Complete Mediation**	Every access to every object must be checked for authorization. Authorization decisions should not be cached. For example, when using JWT tokens for authorization, every action that is performed on behalf of an account on the API should be authorized to its appropriate and limited scope to avoid issues such as Insecure Direct Object Reference (IDOR) and Broken Object Level Authorization (BOLA).	Platform
11	**Token Strength**	If JWT is used, validation of JWT is secured by canceling support to the "none" algorithm. Apply a state-of-the-art encryption algorithm – at the time this document was written, use asymmetric encryption if possible (with at least 512 length keys).	Platform
12	**Input Validation/ Output Encoding**	Service owners should implement appropriate input validation and output encoding practices in all relevant areas in their code.	Service owner
13	**Regex Denial of Service**	When using regular expressions for input validation, be careful not to introduce a denial of service vulnerability for long inputs. Ideally, restrict input length before checking the regular expression.	Service owner/ DevOps

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
14	**Data Sanitation**	Understand what type of data is returned by the API and remove any unnecessary data that is classified as sensitive to the organization.	Service owner/ DevOps
15	**Error Handling**	When an invalid request is sent by the external party, a generic error should be returned which does not disclose any information that may be useful for the attacker (e.g., internal path, filenames, data, etc.).	Service owner
16	**Protection of Testing/Staging Environments**	For many reasons, testing or staging environments are set up insecurely. This may include exposed interfaces, old versions of software, misconfigurations, leftovers, and more. It is highly important to verify that these environments are well protected, preferably inaccessible to external parties and tested for security occasionally.	DevOps
	Phase 3: Testing		
17	**Penetration Testing and Continuous Vulnerability Scanning**	When applicable, perform penetration testing in order to detect and mitigate vulnerabilities. Perform this testing periodically as the attack surface grows/changes, according to the organization's security policy. Penetration tests should be performed regularly by certified professionals using different methodologies in order to increase assurance. It is especially crucial to ensure critical vulnerabilities are addressed before going live.	Security
18	**Security- Oriented Code Review**	Service owners will perform a code review with a security mindset and guidelines based on organizational policies, best practices, and internal application security guidelines.	Service owner/ security

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
19	**Application Security Scanning and Secret Scanning**	Prior to code deployment, it is recommended to scan the code using a Static Application Security Testing (SAST) tool for the existence of application-related vulnerabilities. Additionally, the code should be scanned for any sensitive strings hard coded, such as secrets, passwords, and keys. Any sensitive string must be removed from the code. After deployment, it is recommended to scan the code using a Dynamic Application Security Testing (DAST) tool.	Security
	Phase 4: Implementation		
20	**Valid TLS Certificate**	Install a valid TLS certificate that is up-to-date and signed, according to the organization's security policy, on the API server/API gateway.	DevOps
21	**SSL/TLS Cipher Suites**	Enforce the usage of TLS and only allow strong cipher suites (cancel deprecated and insecure cipher suites), according to the organization's security policy and best practices that will be used when communicating with an API.	DevOps
22	**Source IP Limitation**	Limit the source IP addresses/range of clients reaching the API as much as possible, according to the security risk level. Provide a dedicated IP range for your API consumers.	DevOps
23	**Account Lockdown**	Accounts not in use, unsuccessful attempts to authenticate with invalid credentials, and accounts attempting malicious actions should be locked once a predefined threshold was met.	Platform
24	**Exposed Network Interfaces**	External facing infrastructure that is related to the API hosting should be examined for network interfaces with open ports exposed to the internet. Only mandatory ports should be exposed. Watch out especially for admin APIs.	DevOps

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
25	**Web/API Application Firewall**	Consider protecting the API gateway by an effective web application firewall (WAF) that is designed for protecting API connectivity (not all WAF products are designed to handle API's connectivity).	DevOps/ security
26	**Session Termination**	Inactive authentication tokens or API keys should be terminated following a predefined inactivity time. This logic should be configured on the API GW or using the identity provider platform for the API.	Service owner/ platform
27	**OS/Packages Updates**	The server's OS, installed packages, libraries and third-party software should be updated to the latest version, on an ongoing basis, according to the organization's security policy. Service owners will need to double check that external facing services are updated regularly.	Service owner/ DevOps
28	**Denial of Service Mitigation**	Ensure that API is protected against excessive usage and that resources cannot be exhausted and the API service cannot be disrupted or degraded. The API platform must be protected from denial of service or brute force attacks.	Service owner/ DevOps
	Phase 5: Logging and Monitoring		
29	**Monitoring Across the Entire Stack**	Continuously monitor across the entire API stack for security issues as well as predefined scenarios across the infrastructure, network, and the API functioning and configurations, orchestration, application, and services. Enable audit, access, and other relevant types of logs. Support SIEM integration when applicable.	DevOps
30	**Storing HTTP Access Logs**	HTTP access logs should be stored in dedicated storage to be used in case of an incident or for the cause of digital forensics. The stored HTTP access logs must be retained according to the organization's data retention policy and comply with applicable laws and regulations. Avoid writing and keeping any sensitive data in the logs.	DevOps

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
31	**API Security Tool for Implementing API Protection**	Implement and configure an API security tool that provides visibility over all existing APIs, detection of security incidents related to APIs, response to API security incidents, and additional capabilities that will support detection and response to API-related incidents.	Security
32	**Documentation**	Proper, updated documentation and asset management of the API inventory will greatly ease maintainability and help developers minimize errors and become productive faster. There are many tools, such as OpenAPI Specification, that keep your API documentation updated as your API evolves.	Service owner

Section 2: Design for Egress API Connectivity

This section elaborates on the use case of read/write **to an external entity** (egress connection).

The difference between this section (egress connectivity) and the previous section (ingress connectivity) is that ingress connectivity usually requires more controls, since most of the responsibilities fall under the receiving organization, while in egress connectivity most of the controls are carried out by the receiving side.

Please note that when sharing organization sensitive information with a third party, a risk management process and audit should be part of the design. The goal of the process is to evaluate the security capabilities of the receiving party and which controls should be requested (i.e., when sharing PII, a careful evaluation of the legal terms and the adherence of the third party to relevant privacy laws is required).

Please refer to the suggested controls on Item #6 "Sensitivity of Data" in Part 1: Risk Evaluation.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
		Phase 1: Design	
1	**Threat Modeling and Countermeasures**	After the design phase of egress API connection, a threat modeling session should be performed to enumerate possible threats. A threat model document that summarizes the aforementioned threats should be produced. Any identified threats must be treated with appropriate countermeasures.	Service owner/ security
		Phase 2: Development	
2	**Storing Secrets (Digital Safe)**	Encryption keys and secrets that are used by the service should be stored in a secure location. Access keys and other secrets should be rotated periodically according to the organization's security policy.	Service owner/ DevOps
		Phase 3: Testing	
3	**Application Security Scanning and Secrets Scanning**	Prior to code deployment, it is recommended to scan the code using a Static Application Security Testing (SAST) tool for the existence of application-related vulnerabilities. Additionally, the code should be scanned for any sensitive strings hard coded, such as secrets, passwords, and keys. Any sensitive strings must be removed from the code. After deployment, it is recommended to scan the code using a Dynamic Application Security Testing (DAST) tool.	DevOps/ security
		Phase 4: Implementation	
4	**TLS Valid Certificate**	When establishing the connection to the third party, a TLS certificate validation should be performed. If the certificate is broken/invalid, terminate the connection.	DevOps/IT
5	**Session Termination**	Inactive external connections should be terminated. Service owners should define the business logic.	Service owner

© Copyright 2021, Cloud Security Alliance. All rights reserved.

#	Control	Description	Suggested Owner
6	**Destination IP and Port Limitation**	Network connections to external third parties should be limited to specific IP addresses and ports.	DevOps
		Phase 5: Logging and Monitoring	
7	**Continuous Monitoring**	Monitor the service for local security issues and pre-defined security-oriented scenarios that may cause damage.	Service owner
8	**Detection and Response**	Establish appropriate detection and response capabilities on the service. This includes the implementation of tools and procedures.	Service owner
9	**Documentation**	Proper updated documentation and asset management of the API inventory will greatly ease maintainability and help developers minimize errors and become productive faster. There are many tools, such as OpenAPI Specification, that keep your API doc updated as your API evolves.	Service owner

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Final Thoughts

As the need to connect third parties grows, security professionals are in the best position to provide effective and efficient solutions to achieve the business' needs. As the document describes, there are many scenarios for the mentioned connectivity (ingress, egress, marketplace) and a single document cannot capture all these variations. The focus of this document is to help define the right process, which starts with the idea and goes through design, development, testing, implementation, and monitoring.

Completing the risk evaluation process (Part 1) helps to achieve a better understanding of the integration scope and associated security risks, while the security guidelines (Part 2) provide a drill-down into the tech aspects that should be taken into consideration when defining security controls to manage security risks.

This document may be used as a reference for in-house standards in organizations adopting third-party integration over API. By doing so, organizations will be able to take advantage of API security best practices and recommendations to access the data and services of third-party APIs as well as of other external applications. This reference will help align (and maybe automate) the on-boarding process of new third-party integrations across your organization with a security mindset.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Appendix A: Mapping OWASP API Top Ten to Section 1 Controls

OWASP API Top Ten	Part 2, Section 1 (Ingress API) Control Number
Broken Object Level Authorization	1, 3, 5, 10, 17, 18, 19, 31
Broken User Authentication	1, 2, 3, 8, 9, 11, 17, 18, 23, 26
Excessive Data Exposure	1, 2, 3, 7, 10, 14, 17, 22
Lack of Resources and Rate Limiting	1, 3, 5, 7, 10, 13, 14, 15, 16, 22
Broken Function Level Authorization	1, 5, 8, 10, 16, 17, 18, 19, 30, 31
Mass Assignment	1, 2, 3, 5, 7, 10, 12, 14, 15, 17, 18, 19, 31
Security Misconfiguration	1, 8 , 9, 14, 16, 17, 29, 30
Injection	1, 6, 8, 12, 14, 15, 16, 17, 18, 19, 24, 31
Improper Assets Management	1, 8, 15, 16, 17, 20, 21, 24, 27, 29, 30, 31
Insufficient Logging and Monitoring	1, 15, 17, 29, 30, 31

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Appendix B: Third-Party Integration – Non-API Access Security Guidelines

In previous sections, the document elaborates on integration with third parties via custom API integrations. Those scenarios usually include technical standards (e.g., REST) and authorization logic that ensures a minimum set of boundaries to the integration. There are other scenarios for opening up an application service for third parties in order to exchange data without using custom API access; those are considered in this document as non-API access scenarios. In those scenarios, the connection methods are often less standardized.

Examples for non-API access:

- The HR team wishes to share employee data with a new employee survey vendor by allowing the vendor to get access to the identity system.
- The finance team is required to open an S3 bucket in order to sync its financial reports with the company's auditors.
- A new monitoring tool is requesting exposure of test data.

Because non-API access can cover multiple access methods and a wide range of services, it is challenging to get to a checklist that will cover all scenarios and all controls. Instead of using a checklist, it is suggested to use a list of principles that need to be considered for non-API access.

Connectivity and Access

- Make sure that the transport layer is encrypted.
  - TLS and VPN are valid options.
- Make sure that only relevant ports are opened, and only to approved IP space.
- If applicable, consider applying direct peering with the vendor's infrastructure (AWS, Azure, GCP).

Authentication

- Every access should be authenticated using rotating credentials.
  - STS tokens are good examples and supported by most cloud vendors.
  - In some cloud services, there are ways to increase assurance by adding extra layer of security on top of the token (i.e., use referrer in the bucket policy).

Authorization

- Authorization is usually dependent on the type of services that are accessed (object storage authorization is different than database roles), but in all scenarios, authorization should consider:

© Copyright 2021, Cloud Security Alliance. All rights reserved.

- The least privilege principle.
    - If the access is to a cloud service (i.e., bucket) make sure to create a limited IAM role to this operation.
- Separating the data into authorization silos (different buckets, different DB views).

Ongoing Monitoring

- In many scenarios, preventing abuse of the service can only be controlled using monitoring. (i.e., a service is designed to access one bucket object daily, on a certain day, the service accesses all objects in the bucket. Ongoing monitoring should detect and mark the event as suspicious.

Automation

- Avoid manual work as much as possible. Always prefer to manage your external integrations and any other service via a centrally-managed process (e.g., Terraform, CloudFormation, Chef, Puppet etc.)

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Appendix C: SaaS Marketplace Integration

Another increasingly popular form of third-party integration in modern SaaS environments is through integrations provided on top of the SaaS platform, often made available via a Marketplace or an App Store.

Examples for SaaS marketplace integration:

- Slack: https://slack.com/apps
- Salesforce: https://appexchange.salesforce.com/
- Zendesk: https://www.zendesk.com/apps/

While these integrations may share many similarities with standard API-based integrations, there are a few inherent key differentiators that should be noticed when applying such integrations. These include:

- **Limited Control on Both Sides:** Whereas with traditional API type integrations at least one side is controlled by the organization itself, with marketplace type integrations both sides are implemented by third-party – with one side being the SaaS platform itself, and the other side being the integration/plugin provider. This limits control over the integration, and makes it difficult to create compensating controls.
- **Integrations are Often Provided "As Is":** The integration works in a specific way and asks for specific permissions which more often than not cannot be controlled. This may introduce substantial risk and prevent implementing a least privilege approach.
- **Limited Visibility:** As both sides are provided by third party, any visibility into which actions take place is limited. In most cases, there is no way for the organization to assess the actual interface/communication, and there may be very limited logging capability for detection and response needs.

It is important to note that this document is focused on traditional API-based third-party integrations, and should not serve as a thorough recommendations document for marketplace type integrations.

Nonetheless, due to the increased usage of this type of integration, this document includes some baseline recommendations that should be considered where evaluating such potential integration:

- **Consider Necessity:** With the growing trend of marketplace integration, many plugins today provide functionality that may be achieved in other ways with lower risk. Consider the necessity of each integration before approving such integrations.
- **Consider Provider:** As most control over both sides of the integration is outside the hands of the organization, it is extremely important to assess the providers of both the SaaS platform and the vendor plugin. For instance, allowing an unknown third-party provider to gain full access to your entire Git* repository may introduce unreasonable risk.
- **Review Permissions:** Most marketplaces will indicate what type of permissions are

© Copyright 2021, Cloud Security Alliance. All rights reserved.

required by the integration. Assess that against the objective the integration is supposed to achieve and make sure this is done using least privileges needed. If not, check whether the integration can be configured differently or whether permissions can subsequently be removed after authorizing integration, so that least privileges principles are maintained.

- **Review Marketplace Verification/Certification:** Many marketplaces will offer two or more tiers of integrations, where basic integrations are available for any provider, and "Approved" or "Verified" integrations are available for providers going through a certain certification process. Be extremely cautious with approving unauthorized/uncertified integrations or plugins. Double check the verification/certification terms of service, scope, and limitations (especially regarding privacy and personal data) prior to allowing an integration.
- **Consider All Above Controls:** As much as possible, consider applying the controls specified above for API-based integrations on third-party integrations, including least privilege, security testing (attestation can be requested from the provider), etc.
- **Turn on Logging:** As much as possible, turn on detailed logging on both the SaaS platform and the integration provider to allow breach identification as well as incident investigation if needed.

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Appendix D: Mapping to Other CSA Resources

CCSK

This Document	CCAK Modules
Part 1: Risk Management	CCAK Study Guide Module 1: Cloud Governance
Part 2: Security Checklist	CCAK Study Guide Module 6: Cloud Auditing Module 7: CCM Auditing Controls

© Copyright 2021, Cloud Security Alliance. All rights reserved.

Further Reading

Supply chain risk management
SP 800-161 - Supply Chain Risk Management Practices for Federal Information Systems and Organizations

API security guidelines
OWASP API Security Top 10

API authentication methods
Quick review of API authentication methods

Using STS tokens
What is what is STS service

Understanding threat modeling
https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

Principles of complete mediation
http://owasp-aasvs.readthedocs.io/en/latest/requirement-2.1.html

© Copyright 2021, Cloud Security Alliance. All rights reserved.