How to understand the **NEW!** security requirements table

The new security requirement table will be broken down into the following parts:

- “**Type of requirement**” indicates the requirement category, such as ‘data protection’ or ‘application security.’ This is meant to indicate the security areas or types of security practices that developers should focus on.
- “**Security Requirement**” lists the actual requirement.
- The next four columns – “**Connect**,” “**Forge with data egress**,” “**Forge**,” and “**Power-Ups**” – apply this requirement to these various app types.

In these sections, we list implementation details on how apps **must** or **must not** apply these requirements to their apps.
- As you can see, all “**Forge with data egress**,” “**Forge**,” and “**Power-Ups**” are tagged as **NEW** since we’ve never explicitly listed requirements for these types of apps before.

### Table Key

**Security Requirements**

- **UNCHANGED** = This is an existing security requirement
- **UPDATED** = This is an existing security requirement with new language
- **NEW** = This is a new security requirement
- **FRAMEWORK SUPPORTED** = This requirement is supported by either the ACE or ACSB frameworks provided by Atlassian.

**NOTE:** Framework Supported does not mean that these requirements are automatically met if you are using these frameworks.

**App Types**

- **APPLICABLE** = This security requirement applies to these types of apps
- **PLATFORM PROVIDED** = Atlassian’s Forge platform covers for this security requirement
- **NOT YET APPLICABLE** = This security requirement is not yet applicable to these types of apps.

### NEW! Security Requirements Table

<table>
<thead>
<tr>
<th>TYPE OF REQUIREMENT</th>
<th>SECURITY REQUIREMENT</th>
<th>CONNECT</th>
<th>FORGE WITH DATA EGRESS</th>
<th>FORGE</th>
<th>POWER-UPS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authentication &amp; Authorization</strong></td>
<td>UPDATED</td>
<td>APPLICABLE</td>
<td>APPLICABLE</td>
<td>APPLICABLE</td>
<td>APPLICABLE</td>
</tr>
<tr>
<td>1. An application must authenticate and authorize every request on all endpoints exposed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Anonymous access to application endpoints and resources can be allowed in scenarios where it is needed.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Before making calls using <code>asApp()</code> , you must verify the expected permissions (for example, from product context) with the permissions REST APIs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. An application must default to using <code>asUser()</code> when performing an operation on behalf of the user.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Before making calls using <code>asApp()</code> , you must verify the expected permissions (for example, from product context) with the permissions REST APIs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**NEW!**
An application must always use JWT as the authentication type to validate the application identity and the integrity of the request. Read about implementing authentication here: https://developer.atlassian.com/cloud/confluence/understanding-jwt/

Exception: It is permissible to set authentication type to None in the app descriptor if all of the endpoints in the descriptor serve only static content.

A JWT token must be validated on the server-side for every authenticated request. Validate all user permissions to ensure that only permitted users can execute actions within an application. Read about implementing authorization here: https://developer.atlassian.com/developer-guide/connect-app-authorization/#approaches-for-authorization

A JWT token must be properly validated on the server-side to ensure that the JWT signature is (a) valid, (b) not expired, (c) it comes from your Power-Up, and (d) is a particular Trello user. When creating an API key for use with Trello's REST API, set the appropriate allowed domains for your application. When creating an API key for use with Trello's REST API, set the appropriate allowed domains for your application.
4. **NEW FRAMEWORK SUPPORTED**
   An application must always validate install/uninstall lifecycle requests.

5. **NEW** An application must not accept context JWTs in module or lifecycle endpoints.

---

**Data Protection**

1. **UPDATED** Any Atlassian End User Data:
   - stored by an application outside of the Atlassian product or users' browser must ensure full disk encryption at-rest.
   - accessed by an application or a service should be authenticated and authorized appropriately.

An application must use TLS version 1.2 (or higher) to encrypt all of its traffic, and enable HSTS with a minimum age of one year.

We highly recommend only allowing clients to connect using ciphersuites listed under the "Intermediate compatibility" section of the latest Mozilla’s Server Side TLS guidance. These ciphersuites provide a good balance of security and compatibility with older clients. If you choose to only implement a subset of the ciphersuites, you should thoroughly test any production changes to avoid customer impact.

An application must follow the “Principle of Least Privilege,” when requesting app scopes. This means that an application should only request scopes required to perform its intended functionality, and nothing more.

When making use of `t.set()` from the Power-Up client library, the `shared` scope should never be used to store data that should be kept secret. Please note that data stored in the `shared` scope is readable by any user who can view the board.

---

### New

1. An application must follow the "Principle of Least Privilege," when requesting app scopes. This means that an application should only request scopes required to perform its intended functionality, and nothing more.

   Read about Connect scopes here: [https://developer.atlassian.com/cloud/jira/platform/scopes-for-connect-apps/](https://developer.atlassian.com/cloud/jira/platform/scopes-for-connect-apps/)


**Application Security**

<table>
<thead>
<tr>
<th>Updated</th>
<th>Applicable</th>
<th>Platform Provided</th>
<th>Platform Provided</th>
<th>Applicable</th>
</tr>
</thead>
</table>
| 1. An application must securely store and manage secrets, which include JWTs, OAuth tokens, Trello tokens, sharedSecret, API keys, and encryption keys. They cannot be stored in places that are easily accessible. Example of places include but not limited to:  
  - Source code  
  - URL strings  
  - Referer headers  
  - Application logs  
  - Code repositories, such as Bitbucket and Github |  |  |  |  |

**Applicable**

1. **NEW** An application must never store secrets or authorization information in Entity properties / Content Properties.

**Platform Provided**

1. Use encrypted environment variables and storeSecret to store secrets in your app.

**Platform Provided**

1. Use encrypted environment variables and storeSecret to store secrets in your app.

**Applicable**

1. Use t.setSecret to store secrets such as oauth token.

Read about secure data storage with Power-Ups here: https://developer.atlassian.com/cloud/trello/power-ups/client-library/managing-secrets/#t-storesecret-key--data-

**Application Security**

<table>
<thead>
<tr>
<th>Updated</th>
<th>Applicable</th>
<th>Not Applicable</th>
<th>Applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. An application must maintain and securely configure domains where the application is hosted.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Applicable**

The following needs to be true for the baseUrl listed in an application’s app descriptor:

1. **UNCHANGED** An application must maintain control of each domain.
2. **NEW** An application owner must maintain valid TLS certificates of the domains where an application is hosted, and the domain must be signed by a trusted Certificate Authority.
3. **UNCHANGED** An application’s DNS configuration for subdomains must reference services that are in use.

**Applicable**

The following needs to be true when an application makes calls to domains owned by the application owner:

1. An application must maintain control of each domain.
2. An application owner must maintain valid TLS certificates of the domains where an application is hosted, and the domain must be signed by a trusted Certificate Authority.
3. An application’s DNS configuration for subdomains must reference services that are in use.
<table>
<thead>
<tr>
<th>UPDATED</th>
<th>APPLICABLE</th>
<th>PLATFORM PROVIDED</th>
<th>PLATFORM PROVIDED</th>
<th>APPLICABLE</th>
</tr>
</thead>
</table>
| 1. When applicable, an application must enable security headers and cookie security attributes. | 1. **NEW** An application must set Content Security Policy (CSP) header. Content-Security-Policy: script-src <source>
   
a. Do not use unsafe-inline or unsafe-eval directives in script-src when possible. This will make the policy ineffective against XSS vulnerabilities. |  |  |
| | 2. **UNCHANGED** An application must implement the Referrer-Policy header.
   
a. The header must not be configured to no-referrer-when-downgrade or unsafe-url. We recommend using no-referrer or strict-origin-when-cross-origin. |  |  |
| | 3. **UNCHANGED** An application must disable caching on all HTTPS pages that contain sensitive data by using "no-cache" and "no-store" instead of "private" in the cache control header. |  |  |
| | 4. **UNCHANGED** For session-related cookies, an application must set HttpOnly and Secure attributes when sending Set-Cookie headers. |  |  |
1. An application must validate and sanitize all untrusted data and treat all user input as unsafe to mitigate injection-related vulnerabilities.

Untrusted data is any input that can be manipulated to contain a web attack payload.

2. **NEW** An application using template engines must not use dangerous functions or modules that lead to arbitrary code execution. When there is a business requirement to use these functions, sandbox it inside an isolated, locked-down environment.

---

1. You may either use provided `safe(rawHTML)` or your framework of choice to sanitize the content before insertion into the DOM.

---

1. An application must not use versions of third-party libraries and dependencies with known critical or high vulnerabilities. When vulnerabilities in these libraries and dependencies are discovered, an application owner must remediate them as quickly as possible.
<table>
<thead>
<tr>
<th>Privacy</th>
<th>10. An application must not collect or store credentials belonging to Atlassian user accounts such as user passwords or user API tokens. If an app is currently requesting or storing Atlassian API tokens in order to access a REST API that does not currently support authentication from apps, the app developer can notify Atlassian and receive a temporary waiver for this requirement while Atlassian makes the proper changes to our API to support authenticated requests from apps. Once the API supports approved authentication methods, the app developer will be given a reasonable amount of time to migrate away from using Atlassian API tokens. This requirement does not prohibit apps from storing credentials used to access non-Atlassian applications.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>1. NEW If the security incident involves sharedSecret leakage, immediately notify Atlassian and request to rotate the sharedSecret through DEVHELP within 24 hours.</td>
</tr>
</tbody>
</table>
13. You must identify at least one email as a security contact and have them create an account on ecosystem.atlassian.net so that they are notified about vulnerabilities in the app via Atlassian Marketplace Security (AMS) tickets.

The following guide explains how to get access to AMS: [https://developer.atlassian.com/platform/marketplace/vulnerability-review-practices-for-atlassian-partners/](https://developer.atlassian.com/platform/marketplace/vulnerability-review-practices-for-atlassian-partners/)

Please note that an admin can also be listed as a security contact.