Atlas Threat Model

Table of Contents

[Introduction 2](#_Toc32215137)

[Code Repos 2](#_Toc32215138)

[Documentation 2](#_Toc32215139)

[Data Flow Diagram 3](#_Toc32215140)

[Related Services or Applications 3](#_Toc32215141)

[Attack Surface/Adversary View 4](#_Toc32215142)

[Entry points 4](#_Toc32215143)

[Assets 4](#_Toc32215144)

[Twitch Data Classification 4](#_Toc32215145)

[Trust Levels 4](#_Toc32215146)

[Common use cases and scenarios 5](#_Toc32215147)

[Threats and Attacks 5](#_Toc32215148)

[Notes 7](#_Toc32215149)

[Implementation Assumptions 7](#_Toc32215150)

[External Security Notes 7](#_Toc32215151)

[Internal Security Notes 7](#_Toc32215152)

[Past Issues 8](#_Toc32215153)

[Glossary and Nomenclature 8](#_Toc32215154)

# Introduction

Atlas is a Premium Content Catalog designed to improve internal operational efficiencies, power tech-enabled solutions, and increase the overall visibility of upcoming events for the benefit of viewers, creators and employees alike.

## Code Repos

* Anchor (Front End)
  + Content Tech Team: <https://github.com/crsdata/Anchor>
  + Github Enterprise: <https://git-aws.internal.justin.tv/content-tech/Anchor>
* Resonance (Back end)
  + Content Tech Team: <https://github.com/crsdata/resonance-api-server>
  + Github Enterprise: <https://git-aws.internal.justin.tv/content-tech/Resonance>

## Documentation

* Anchor - <https://wiki.twitch.com/pages/viewpage.action?pageId=215393645>
* Resonance - <https://wiki.twitch.com/display/CT/Resonance>

# Data Flow Diagram

* Atlas Overview (TBD): [Lucid Chart](https://www.lucidchart.com/invitations/accept/4bc7f8db-4d0b-46f3-896a-ed2c4d81bd91)

Process Flows

* Auth: [Lucid Chart](https://www.lucidchart.com/invitations/accept/874de8b5-9df5-4fb6-bbb3-64328a3b206c)
* Asset Uploading: [Lucid Chart](https://www.lucidchart.com/invitations/accept/d0de5d36-8ae7-4f3d-9031-e6deec55b92b)

Endpoints

* Anchor: <https://wiki.twitch.com/pages/viewpage.action?pageId=215393645>
* Auth: <https://wiki.twitch.com/display/CT/Auth+Service>

|  |
| --- |
| Related Services or Applications |
| (TBD?) Twitch Upload Service |

## 

# Attack Surface/Adversary View

A system can only be attacked if it has entry points. Entry points are where commands or data crosses the boundary of the application. Furthermore, an adversary will attack a system only if that system has one or more assets of value.

|  |
| --- |
| Entry points |
| *Atlas Service* |
| *Auth Service* |
| *Network access to a database* |
| *React Front end* |
| *(TBD) Server Serving Front end* |
| *ECS / Fargate* |
| *Nginx Docker* |

## 

|  |  |
| --- | --- |
| Assets | [Twitch Data Classification](https://wiki.twitch.com/display/SEC/Twitch+Data+Classification+and+Handling+Policy) |
| *Upcoming Unannounced Events* | *?* |
| *RDS credentials, LDAP Credentials* | *Customer Data* |
| *Availability (your application should in theory always be up and running)* | *N/A* |
| *1 Web server for Frontend, 1 Web server for Backend, 1 Aurora DB Server* | *N/A* |

|  |
| --- |
| Trust Levels |
| *Unauthenticated unauthorized user - All web pages other than /login require authentication* |
| *Authenticated, unauthorized user – Any user who successfully authenticates with SSO, but is not part of the LDAP groups or a holder of provisioned API tokens that allow access requested assets.* |
| *Authenticated, authorized user - Any user who can successfully auth with SSO and is part of authorized LDAP groups or a holder of provisioned API tokens.* |
| *Authenticated admin - Any user part of an LDAP group that gives access to elevated permissions* |

|  |
| --- |
| Common use cases and scenarios |
| Product Manager: A user who can create products on the platform which allows management of all product features. These include managing product assets, events, streams, and seasons. |
| Event Manager: A user who can create events on the platform, and manage details related to those events. |
| Twitch Account Manager: A user who can create and manage streams and events for channels they own. |
| Exporting User: A user who can export query data to csv format |
| Browsing User: A user who can query active events |

|  |
| --- |
| Threats and Attacks |
| Attackers may attempt to frame the Atlas web site in order to perform “clickjacking” attacks against legitimate staff who are logged into it and who are also visiting “evil” web sites in another browser window. Mitigate this with X-Frame-Options: DENY or the equivalent Frame Ancestors Policy in CSP. More information about Clickjacking is available on the [Developer Resources](https://wiki.twitch.com/display/SEC/Developer+Resources) page. |
| Attackers may attempt to cause a staff member’s browser to emit any authorization tokens session cookies used by this application in clear text across the internet, e.g. by using an evil web site to issue a redirect to a staff member’s browser to the plain <http://atlas> url. Mitigate this by using the “secure” flag on all such cookies and limiting the site to only use HTTPS. More information about the “secure” cookie flag is available on the [Developer Resources](https://wiki.twitch.com/display/SEC/Developer+Resources) page. |
| Attackers may attempt to hijack a staff member’s logged in session in Atlas through Cross Site Scripting (XSS) attacks. They can attempt this despite not having direct access to the application, because they can formulate links and redirects that cause a staff member’s browser to visit Atlas with e.g. unexpected parameters. See the XSS materials on our [Developer Resources](https://wiki.twitch.com/display/SEC/Developer+Resources) page for information on avoiding this. |
| Attackers may attempt to obtain, alter, or destroy the information in the application, or cause code execution on our service, using SQL Injection attacks. They can attempt this despite not having direct access to the application because they can formulate links and redirects that cause a staff member’s browser to visit Atlas with e.g. unexpected parameters. See the Injection materials on our [Developer Resources](https://wiki.twitch.com/display/SEC/Developer+Resources) page for information on avoiding this. |
| Attackers may attempt to abuse API’s exposed by the API Service, taking advantage of any places where authorization is not required and where a staff member’s browser can be instructed to make requests against the Atlas api initiated from an evil web site. Mitigate this by requiring a “non-ambient” form of authorization (e.g. explicit Authorization header with bearer token) on all API requests and avoid overly permissive CORS policies (e.g. there should be no reason to use Access-Control-Allow-Origin: \* given the intended use of this application, you can permit only the front end web server’s origin). For more information see our [Developer Resources](https://wiki.twitch.com/display/SEC/Developer+Resources) pages for CSRF and CORS. Following these instructions should avoid the need to pursue the traditional “CSRF Token” mitigation strategy required in “ambient auth” situations. (The “ambient” vs “non-ambient” distinction is defined on our CSRF wiki page.) |
| Staff members may use access to this service for inappropriate purposes. Mitigate this risk and comply with relevant Data Handling Policy provisions by limiting access to persons with a need to know (“Restricted Access”) and log activity as described in <https://wiki.twitch.com/display/SEC/Twitch+Data+Handling+Standard> and <https://wiki.twitch.com/display/SEC/Logging+Best+Practices> |
| Attackers who are positioned to inspect internet traffic will attempt to take advantage of any unencrypted communications if they were to occur (e.g. to Salesforce). Ensure transport encryption is used (as is also required in Security Expectations). |
| Attackers would like to obtain secrets needed to connect to Atlas et al. Employee laptops occasionally get compromised, stolen, etc and can compromise secrets if they are stored along with source code on those laptops. Using secrets management (e.g. KMS) is a best practice to mitigate this risk. |
| Attackers will attempt to connect directly to services like S3. Ensure permissions are configured appropriately (ie, not world-accessible). |

# 

# Notes

|  |
| --- |
| Implementation Assumptions |
| *We assume that anyone who has access to Twitch SSO is a valid employee* |
|  |

### 

|  |
| --- |
| External Security Notes |
| Authentication accepts either a jwt token, or API token for authorization. This allows us to support both internal and external users as required. |
| Auth is controlled using JWT tokens with a 1 month lifespan. Users store secure cookies containing this token for communication with backend services. |
| We handle data at Customer and Restricted classifications |
| Secret keys for databases are stored in S3 encrypted with KMS |
| CORS support is enabled |
| We want to develop a revocation plan for our tokens so we can support handling a user having their token be compromised |

### 

|  |
| --- |
| Internal Security Notes |
| We use .net core for the backend service |

|  |
| --- |
| Past Issues |
| *Description of a past issue + a Jira link* |
|  |

|  |
| --- |
| Glossary and Nomenclature |
| AM: Account Manager |

## 