# 

Amp Threat Model

Table of Contents

[**Data Flow Diagram**](#_d49fwh48ajlu) **2**

[Related Services or Applications](#_ojdxmi3p7jk3) 2

[**Attack Surface/Adversary View**](#_48cgcrli0uu) **3**

[Entry points](#_ayynx22ihe2e) 3

[Assets](#_okesmygetjiz) 3

[Twitch Data Classification](#_rtk46wsoch4y) 3

[Trust Levels](#_b1sfn7qj9nfd) 4

[**Common use cases and scenarios**](#_44kwj66ahea6) **4**

[**Threats and Attacks**](#_68k69o5xj9mc) **5**

[**Notes**](#_ktshjldw215z) **6**

[Implementation Assumptions](#_uwyw4mub9ohl) 6

[External Security Notes](#_y8g7av9a570l) 6

[Internal Security Notes](#_97axoxhjydq) 6

[**Past Issues**](#_bqyaem3o2ir8) **6**

[**Glossary and Nomenclature**](#_jcw30cpbme1r) **7**

# Introduction

Amp is a service which supplies aggregated twitch data for helping twitch employees build requests for marketing opportunities, user testing, and sales campaigns. It allows for searching for users to Target and optionally to export their data for external uses. Metadata is attached to the requests and is sent off to Salesforce for account manager s to present to their partners.

## Code Repos

* Ripple(Front end) - <https://github.com/crsdata/Ripple>
* Resonance(Back end) - <https://github.com/crsdata/resonance-api-server>

## Documentation

* Ripple - <https://wiki.twitch.com/pages/viewpage.action?pageId=189280818>
* Resonance - https://wiki.twitch.com/display/CT/Resonance

# Data Flow Diagram

<https://www.lucidchart.com/invitations/accept/ac817035-7797-41da-8ef1-e7697cd4e26b>

Process Flows

Auth - <https://www.lucidchart.com/invitations/accept/874de8b5-9df5-4fb6-bbb3-64328a3b206c>

Data Import - <https://www.lucidchart.com/invitations/accept/3d50b7cf-549f-47d8-8830-708bee8bd1b7>

Browsing/Exporting users - <https://www.lucidchart.com/invitations/accept/4027fc42-ecad-4c5e-bc43-81df5456218d>

Request Creator - https://www.lucidchart.com/invitations/accept/799b8e81-4caa-4efe-a6a6-7cba5b8b3298

Request Manager - https://www.lucidchart.com/invitations/accept/399aa43a-7549-4635-837b-6d00be01fb46

Salesforce - https://www.lucidchart.com/invitations/accept/990b8b43-4021-43c4-8d3b-8161ec4fa856

Auth Service Endpoints - <https://wiki.twitch.com/display/CT/Auth+Service>

Amp Service Endpoints - https://wiki.twitch.com/display/CT/AMP+Data+Service

|  |
| --- |
| Related Services or Applications |
| Tahoe - All data is sourced from tahoe(https://wiki.twitch.com/display/CT/Data+Sources) |
| Salesforce - We send and receive from salesforce in order to present requests to AMs to discuss with partners and to retrieve contact info to present to the requestor |

## 

# 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 |
| *Amp Service* |
| *Auth Service* |
| *Network access to a database* |
| *Salesforce* |
| *React Front end* |
| *Node.JS Server Serving Front end* |

## 

|  |  |
| --- | --- |
| Assets | [Twitch Data Classification](https://wiki.twitch.com/display/SEC/Twitch+Data+Classification+and+Handling+Policy) |
| *Creator performance stats* | *Restricted Data* |
| *Creator email, country code, language code, revenue data* | *Customer Data* |
| *RDS credentials, Salesforce 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, 1 Redshift/Tahoe DB Cluster, Salesforce* | *N/A* |
|  |  |

## 

Your application will likely be accessed by different classes of users. You should be able to map some of them to the actors of your application and capture the ones you do not expect to interact with your application since they will likely be the ones attempting to attack your system.

|  |
| --- |
| Trust Levels |
| *Unauthenticated unauthorized user - All web pages other than /login require authentication* |
| *Authenticated authorized user - Any user who can successfully auth with SSO* |
| *Authenticated admin - Any user part of an LDAP group that gives access to elevated permissions* |
|  |

## 

This list of common scenarios should reflect what each Trust Level can achieve when interacting with the application. It should reflect the use cases defined in the architecture, but should focus on the security aspects (capabilities, permissions).

|  |
| --- |
| Common use cases and scenarios |
| Request Creator - A user who wants to reach out to creators comes to amp, searches for a good list of creators by using filters on the front page. They create a request and send it off to be approved by the Request Manager. Once it has been approved, the request is sent off to Salesforce to get creators to opt in/out. Once the list is finalized, the Request Creator gets access to the contact info of the creators to faciliate the request. They can also rate/comment on the quality |
| Request Manager - A privileged user who can approve and revise requests. When a Request Creator sends the request off for approvel, the Request Manager can opt to approve it, which sends it to salesforce. They are also given the option to revise the request, which entails them making edits and suggesting new/removed creators, that the Request Creator must co-sign before the request goes to Salesforce |
| Browsing Twitch Employee - A user who simply wants to view the data on partners and see the stats. They would use the filters on the front page to explore the data. |
| Exporting Twitch Employee - A user who wants to use our tools to view the data and export it to csv for use elsewhere. |

## 

This phase should be done in cooperation with a Twitch security engineer, ideally your team's [AppSec Relationship Owner](https://wiki.twitch.com/display/SEC/AppSec+Relationship+Owners). For each entry point, you need to determine how an adversary might try to affect an asset. Based on what the asset is, you predict what the adversary would try to do and what their goals would be.

|  |
| --- |
| Threats and Attacks |
| Attackers may attempt to frame the AMP 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 cleartext 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://amp> 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 AMP 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 AMP 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 AMP 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 AMP 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 Salesforce 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

This section documents the information that the developers and users of the system need to be aware of.

|  |
| --- |
| Implementation Assumptions |
| *We assume that anyone who has access to twitch SSO is a valid employee who should have access to our non-Customer level data* |
|  |

### 

|  |
| --- |
| External Security Notes |
| Auth accepts either a jwt token or api key for auth, so we can support other twitch services consuming our data, as well as testing |
| 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 |
| Keys for salesforce and Databases are stored in S3 encrypted with KMS |
| CORS support is enabled, but not currently configured strictly |
| 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 |
| CSRF protection is not yet in, but it is currently ticketed. It shouldn't necessarily be required since we only have json-based endpoints and cors, but doesn't hurt to have it |
| We use .net core for the backend service |

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

|  |
| --- |
| Glossary and Nomenclature |
| Request - An event requiring partner engagement, this could be partaking in a marketing deal, a sales campaign, user testing, or in real life event  AM - Account Managers, the staff who will actually reach out to partners and confirm/deny their interest in requests  Request Manager - Partnerships staff who can approve/revise requests before they go to salesforce |

## 