IT Security Overview

An overview of IT security guidelines, processes, and systems architecture for the Wipster platform

Updated February 2020

Table of contents

1 Introduction

This document discusses security guidelines, policies, design principles, and implementation at Wipster.

1.1 Target audience

This document provides a guide covering the measures we take to ensure Wipster systems and data are secure. The target audience is technical and non-technical professionals of organisations that use, or intend to use, the Wipster platform.

1.2 About Wipster

Wipster is a pre-publishing platform providing the means for engaging video, image and Portable Document Format (PDF) review and approval among teams creating short-form video content, imagery and other creative output. Content is uploaded to the platform, and teams collaborate via commenting directly on the content. Comments become to-do items, making Wipster the system of record for creative projects. Wipster accelerates the creative process, allowing teams to publish more content, faster.

1.3 Who to contact

Direct any questions regarding this document or the Wipster platform to:

Robyn Haugh
VP of Product
robyn@wipster.io
+64 21 022 59496

2 Infrastructure overview

This chapter outlines the network and server infrastructure for Wipster and gives an overview of the application architecture from an IT security perspective.

2.1 Servers and services

Wipster runs on Microsoft's Azure cloud platform. Wipster operates in a multi-tenanted environment with logical separation of instances controlled by credentialed access. Traffic to our servers is restricted using Azure's security controls and on-server IP filters to isolate server environments and to only allow traffic that is necessary for the operation of the platform.

Access to our servers is highly restricted. Customers never have direct access, except through our web application interfaces, which customers use through their web browsers on personal computers or mobile phones. Administrative access in the application is restricted to employees at Wipster, is based on their *@wipster.io* email account, and requires a change to our application to enable. This level of access is granted only to employees who require it to perform their duties.

Security credentials for logging on to virtual machines are never shared between our production, staging, or other environments.

Wipster also uses Amazon's CloudFront content delivery network (CDN) service to speed the delivery of content to users around the world. Video, image, audio and PDF traffic through CloudFront is served only via HTTPS and CloudFront only accepts requests from our Wipster domain as the origin, maintaining our secure environment. Access to CloudFront servers is restricted in the same manner as our Azure servers.

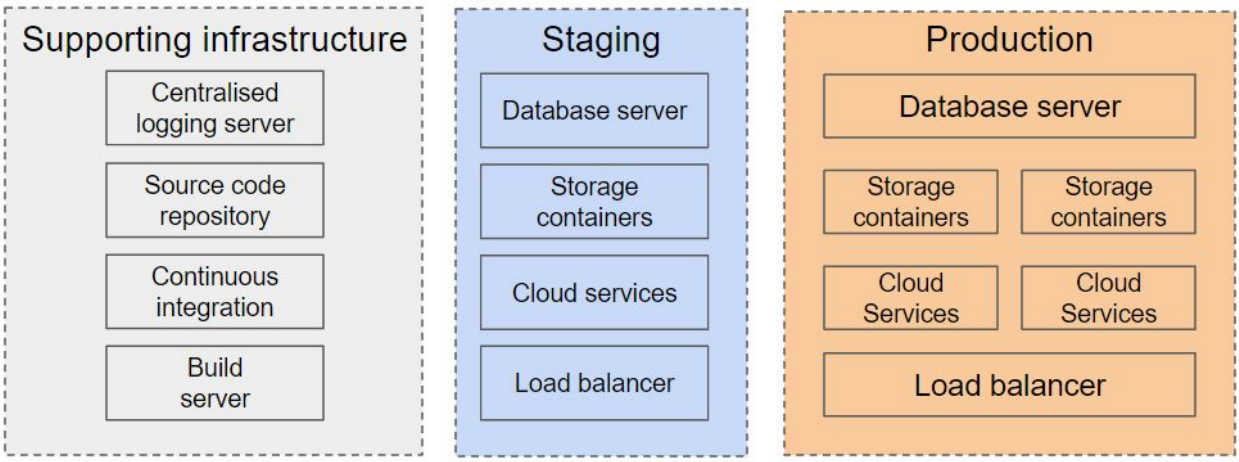
Wipster uses Brightcove Zencoder to encode uploaded videos to two resolutions: one High Definition and one Standard Definition (assuming the resolution of the original upload supports 1080p). These encodings are then used for video playback on the Wipster platform.

2.2 Redundant architecture

Our systems are designed to avoid single points of failure. Video/image/PDF and Audio data and Azure Table (nosql) data are replicated across two Azure datacenters in Geo-Redundant Storage[1]. The cloud services running the Wipster applications are configured to automatically scale up and down in response to demand. The SQL Server database has a standby server in a separate datacenter that will take over automatically if the primary instance goes down.

[1] https://docs.microsoft.com/en-us/azure/storage/storage-redundancy

2.3 Separate security environments



Our servers and services perform different roles and are grouped in environments that are strictly isolated from each other. We have the following broad environments:

- **Development operations infrastructure**
  Each system has its own security and access controls. The infrastructure includes source code control via Git, continuous integration via Microsoft's Visual Studio Team Services, and centralised logging via Seq[2] and Raygun. Access to these systems is strictly limited to employees on a needs-only basis, and served over HTTPS connections.

- **Staging**
  Staging mimics production, but has dedicated cloud services and storage containers, and a dedicated SQL Server instance. Staging is used for end-to-end testing of the Wipster application prior to deploying changes to Production.

- **Production**
  Including database servers, cloud services, storage containers and caching servers.

2.4 Production environments

The Wipster production environment consists of:

- An Azure cloud service-based private API
- Azure cloud services that host the Wipster web and mobile applications

[2] https://getseq.net/

- An Adobe plugin that integrates directly with Adobe Premiere and Adobe After Effects and communicates with the Wipster platform via our private API
- Amazon's CloudFront CDN, used to speed video playback across all apps, and uploads from the the Adobe plugin
- A Redis caching server within our Azure environment powering web notifications
- Several Azure cloud service-based scheduled tasks and worker roles that perform backend functions (such as video encoding), operating via queues

All traffic on the Wipster platform is encrypted via HTTPS:

- From client browser and Adobe software to Azure cloud services and Azure storage containers
- From client browser, through CloudFront, to Azure storage containers for video playback
- From Azure cloud services to the SQL Server database
- From Azure cloud services to Azure Tables (nosql)
- From Azure cloud services to the Redis caching server (port 6380)
- To all third-party services:
    - From Azure storage containers to Zencoder for encoding
    - From cloud services to Braintree[3], our payments provider
    - From cloud services to Auth0[4], our Enterprise Single-Sign On (SSO) provider

The cloud services that run our apps sit behind an Azure load balancer. No direct HTTP(S) access to the cloud services is possible.

Wipster data is encrypted at rest via built-in Azure encryption functionality:

- Video, Image, PDF and Audio data in Azure storage containers
- Azure Table data in storage containers
- SQL Server database (v12.0.2000.8)

Cached data in Redis has a Time To Live (TTL) of 24 hours.

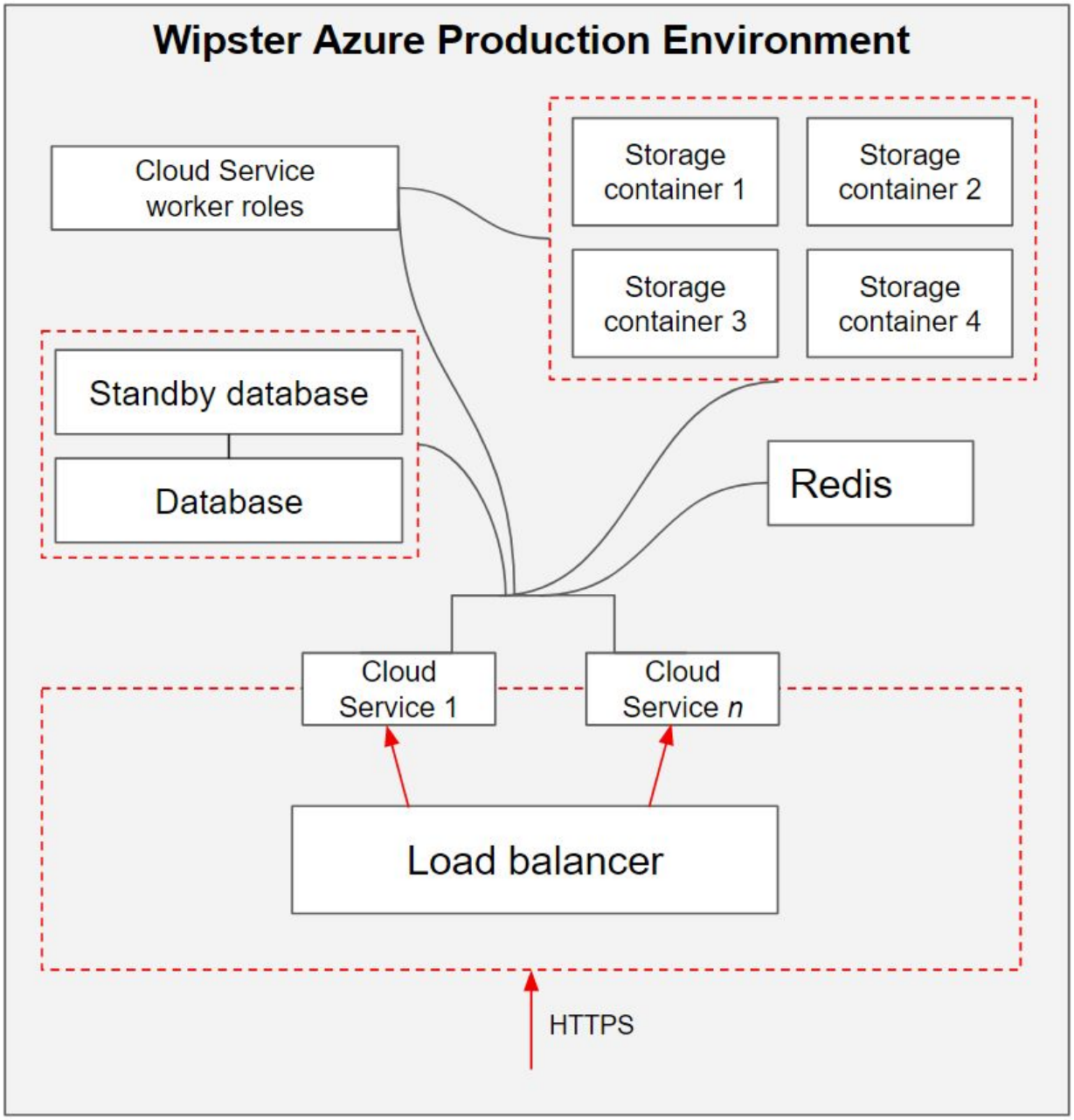
Azure is configured to only allow access to our SQL database to virtual machines within our environment. No other machines and/or IP addresses have access without explicit configuration.

The following diagrams depict:

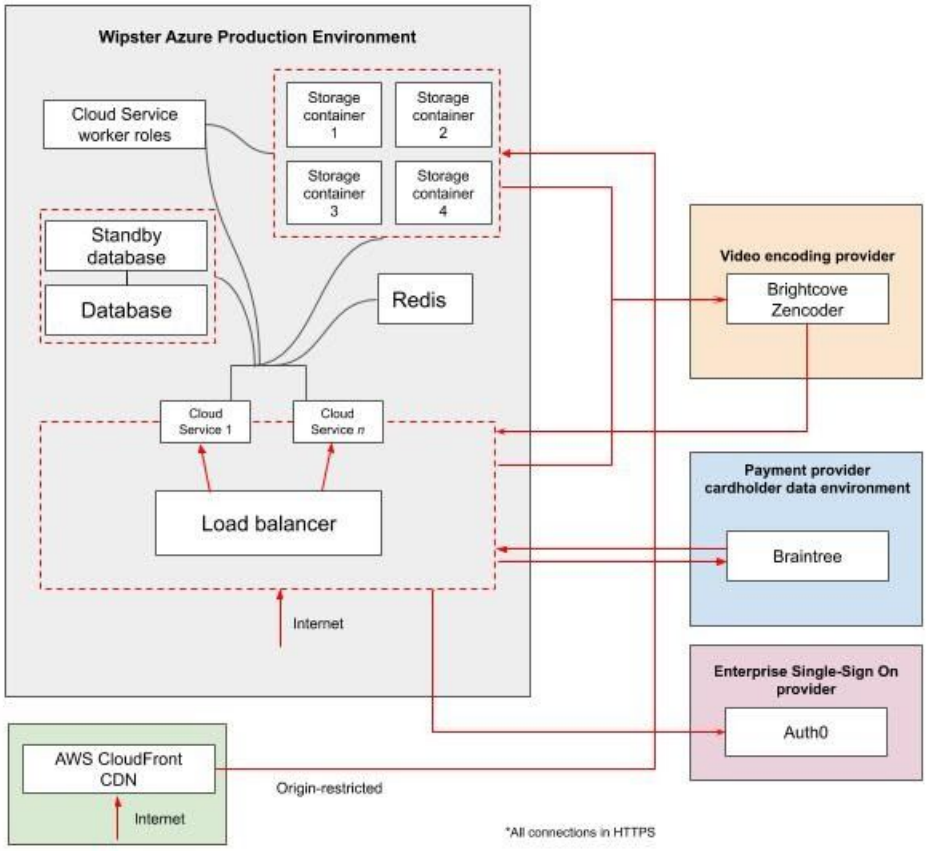
A. Wipster's Production Azure environment
B. Wipster's Production environment with connections to third party systems

[3] https://www.braintreepayments.com/
[4] https://auth0.com/

A.


Wipster Azure Production Environment
Cloud Service worker roles
Storage container 1
Storage container 2
Storage container 3
Storage container 4
Standby database
Database
Redis
Cloud Service 1
Cloud Service n
Load balancer
HTTPS

B.



2.4.1 Wipster Qualys SSL Labs rating

The Wipster app has an “A” rating according to the Qualys SSL Labs SSL Report.[5]

Known weak cryptographic ciphers and protocols have been disabled (e.g. 3DES, SSL2, SSL3, TLS 1.0, TLS1.1 - all disabled)

3 Application architecture

The Wipster platform consists of:

- Microsoft .NET backend application
- Google AngularJS + Microsoft TypeScript single-page web application [link]

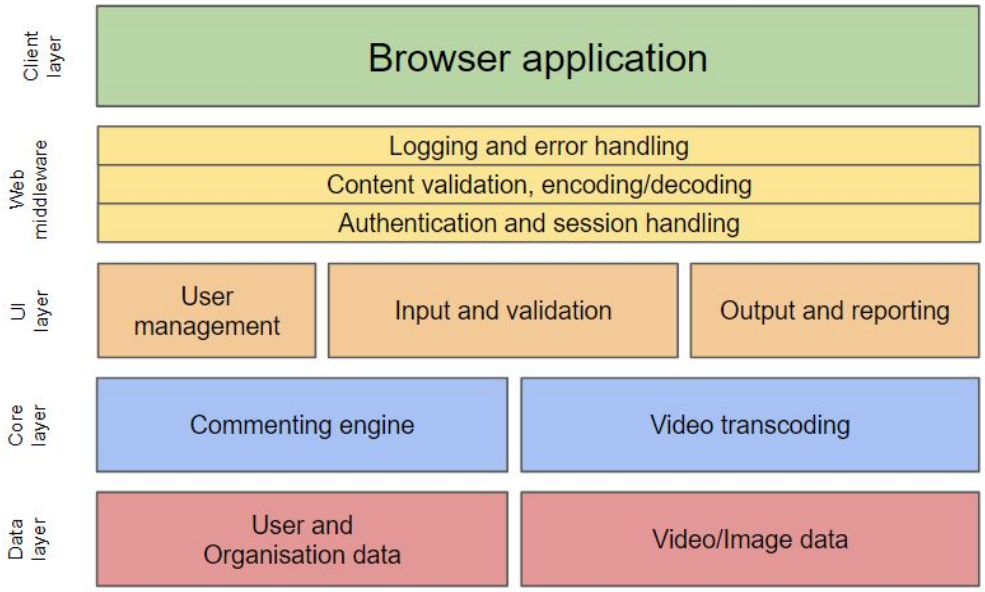
[5] https://www.ssllabs.com/ssltest/analyze.html?d=app.wipster.io

- Google AngularJS + Microsoft TypeScript single-page web application targeting mobile phone browsers [link]
- Google AngularJS application (hereafter panel) that integrates into Adobe Premiere and Adobe After Effects [link]

3.1 Web application logical structure

The web, mobile, and panel applications form the interfaces between the Wipster platform and end users, all served via HTTPS.

A comprehensive stack of web middleware (mainly Microsoft's Open Web Interface for .Net [OWIN][6] and Microsoft's WebApi2[7]) handles communications between the server application and end users. It is responsible for URL routing, content validation, shared access token handling, session handling, and for mitigating common web application vulnerabilities.



3.2 Permissions model

Immediately upon visiting the web, mobile and panel applications, users are required to authenticate. Users can have an account directly within the Wipster application, or can choose to login in via a federated identity from:

- Google [API link]

[6] https://github.com/TerribleDev/OwinOAuthProviders
[7] https://msdn.microsoft.com/en-us/library/dn448365(v=vs.118).aspx

- Facebook [API link]
- Vimeo [API link]
- Enterprise Single Sign-On via Auth0 [API link] (more detail below)

Authorisation occurs deeper in the application and determines whether a given user can perform a specific action.

This authorisation operates on two levels:

- The backend application code restricts actions to specific system roles using .NET's in-built Authorisation attribute decorator. This attribute is attached to specific methods/controllers, and grants access to all or a subset of roles as required. This is used to control Wipster employee administrative access vs. standard external user access. As covered in this document, only a defined list of Wipster employees, based on their *@wipster.io* email address, can access administrative functionality, and the attribute decorator is used to ensure this is enforced.
- The backend code enforces application access based on various roles (see section 3.2.5) using a code object for "Org Role Permissions". This object is present for all authenticated members, and is interrogated for the required permissions when users attempt to access features.

When an authorisation failure occurs, the error is caught by the application and the user is redirected back to their Wipster home screen.

3.2.1 Organisations

The highest level of grouping in the Wipster domain model is the organisation (hereafter org). Everything else belongs to, or is contained within, an org. Users belong to an organisation, and they will never have access to resources outside of their organisation, unless the owner of a resource explicitly grants this access.

3.2.2 Users

Users are identified by a unique email address, and they log in using a password (either locally or via a federated account as covered in section 3.2). Details on local account password management is covered later. Users always belong to an org.

3.2.3 Folders

An org can have one or many folders, each of which generally represent a work project. Folders contain video, image, and PDF content. Access to folders and their contents are controlled by the role the given user has (roles covered in section 3.2.5)

3.2.4 Videos, Images, PDFs and Audio files

Uploaded videos, images, PDFs and audio files reside within folders. Access to these assets is controlled by the role of the given user. Wipster accepts a defined list of file formats, as covered in section 3.4.3.

3.2.5 Roles

Wipster operates with a number of authorised roles that allow different levels of access:

- **Team Owner**
  Full rights to all content, billing, and user management within an org

- **Team Admin**
  Full rights to all content and user management, and read-only access to billing statements in an org

- **Full**
  Access to all team folders and content within an org

- **Guest**
  Access to a specified list of team folders and content within an org

- **Reviewer**
  Access to a specific piece of content via invite (private or public share, as covered in section 3.3.4) within an org

3.3 Application security

3.3.1 Authentication

3.3.1.1 Account credentials

End-user authentication is based on a combination of email address and password. An end-user's email address is used in account creation and password reset procedures, ensuring that the user has control over the given email address. Password protocol applies and they must consist of a minimum of six characters, at least one uppercase and one number.

Passwords are not stored in plain text. The Wipster application makes use of *bcrypt* hashes (provided by a *bcrypt* library integrated into the backend code)[8] specifically designed as a one-way algorithm for password encryption.

[8] https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf

Users are responsible for selecting a sufficiently strong password. The Wipster application assists users in this regard via the AngularJS third-party plug-in *ng-password-strength*[9], which moves from red to yellow to green as password complexity increases.

The entire Wipster application runs via HTTPS, so password data is never transmitted as plain text. Similarly, passwords are not put into log files on our platform.

3.3.1.2 Account creation and password reset

Passwords are not stored as plain text, nor are they sent via email. Once a password is lost, a user can only regain access by initiating a password reset request. The user enters their email address, and if the address exists an email will be generated containing a link to reset their password. The link contains a cryptographically strong token to verify that the password reset request originated from the email Wipster sent to the user. Clicking the link takes the user to a screen where they can create a new password.

3.3.1.3 Enterprise Single-Sign On (SSO)

Enterprise customers can optionally have SSO enabled for their accounts. This allows users to login with the same credentials they use for their internal corporate systems. Enterprise customers can then control password rules and ensure that a dismissed employee's Wipster access will be immediately revoked.

Wipster uses the third-party provider Auth0 to supply enterprise single-sign on services. All communications with Auth0 are done over HTTPS. Once a user successfully logs in via their own corporate login page, Auth0 mediates the transfer of the user back to Wipster along with a token verifying the successful login and identifying information. With this valid token, Wipster matches the email address to an active user in the Wipster database and logs the user in. Once logged in, the user is subject to the same session management that applies to local account logins.

3.3.2 Shared Access Signatures

Once a piece of content is uploaded to Wipster, it can only be accessed through our application. This is ensured via issuing a Shared Access Signature (SAS) for each piece of content inside Wipster. A SAS is a URI that encompasses in its response all of the information necessary for authenticated access to that particular piece of content. This allows Wipster to grant access to a particular piece of content only to authorised parties (e.g. team members), and only for a limited amount of time. For example, if someone were to attempt to circumvent Wipster to gain access to a piece of content directly from our Azure storage containers, they would be refused access as we did not issue a secure token for the resource.

[9] https://github.com/subarroca/ng-password-strength

3.3.3 Session handling

After a successful authentication attempt using an email address/password or via a federated ID, users are issued an OAUTH2 bearer token via OWIN middleware, which is stored in Local Storage on the browser. The token tells Wipster who the user is, what team they are in, and what they can see and do. Every single action inside Wipster is signed with this unique, tamper-proof token. These bearer tokens expire after a period of 24 hours.

3.3.4 Private and public sharing

Content on Wipster can be shared for review via:

- Creating a tokenized link - anyone with the link can view and review; or
- Inviting via email address - the recipient receives an email with a tokenized link

Owners requiring more security can choose to enable private-only sharing for their team. This allows for sharing *only* via email address, and *only* for members of the given team. For both private and public share links, a password can be applied for enhanced security. This requires either link holders or team members to enter a password in order to view the resource.

3.3.5 Third-party integrations

Wipster offers its users the ability to connect their Wipster account to the following third-party services via public APIs:

Communications:

- Slack

Content publishing:

- Brightcove
- Wistia
- Facebook
- YouTube
- Twitter
- Vimeo

Communications with all third party applications are done via HTTPS. Wipster users grant access to third party applications within the Wipster web app. Where possible, this access is granted via three-legged OAuth[10]. Where this is not possible, users are required to manually input access tokens and/or secrets, obtained from their accounts on the given third-party provider

[10] http://oauthbible.com/#oauth-10a-three-legged

3.4 Technical application security measures

3.4.1 Information containment

3.4.1.1 Accounts

Efforts are taken to not disclose information about existing user accounts via using generic messaging during invalid login attempts and elsewhere.

3.4.1.2 API access

The private server API (which is called by the user's browser to access our system) uses symbolic names for the API functions that contain no information about the internal code structure of the application, i.e. no namespaces, module names or function names are leaked to the outside world. URLs are validated and the application and web server provide no direct access to the file system, which makes directory guessing or directory traversal impossible. All API functions are treated as stateless and have security controls that are appropriate. This means that all API requests are treated as untrusted and could be called outside the normal flow of the application. Access to any resource is subject to the platform's authorisation mechanism, which only grants access if explicitly allowed.

3.4.1.3 Error handling

All API calls that result in an error will return an appropriate HTTP status code, and a human-readable error message.

Recoverable errors are handled appropriately, on an application level. E.g. when an authenticated user sends a request that requires authentication, an HTTP 401 error is sent back to the client application, which causes a prompt to the user to log in before retrying the request.

When an unexpected error occurs on the server, an HTTP 5xx status code is returned. The user will see a generic error page informing them something went wrong. It does not display any technical information that could reveal information about the underlying platform to the user.

A global error handler exists in the Wipster platform's application stack, which will catch any unhandled exception before returning a response to the client. Stack traces or other technical data are never displayed to the user.

3.4.2 Administrative access

As covered in section 2.1, administrative access in the application is restricted to employees at Wipster, is based on their *@wipster.io* email account, and requires a change to our application to enable. This level of access is granted only to employees who require it to perform their duties.

3.4.3 Input validation

Wipster has a defined list of file formats that it accepts. All file types not included in these lists are rejected:

Video

- .mp4
- .m4v
- .mov
- .avi
- .mkv
- .mpg
- .mpeg
- .ts
- .m2ts
- .webm
- .wmv

Image

- .jpg
- .jpeg
- .gif
- .png

Document

- .pdf

Audio

- .mp3
- .wav
- .m4a

Users can submit text-based input via comments and replies on videos and images, naming videos and images, and searching for content in their accounts. Validation is done on both the front and back ends to ensure the data we get is the data we expect.

3.4.4 Output escaping

As part of our development process, every piece of information that can be rendered in the user's web browser is checked to ensure it will not be shown as unfiltered or unescaped output.

Information is either generated by the application (e.g. static content or strings that are hard-coded in the application) or is retrieved from persistent storage or user input.

The web and mobile browser applications never renders strings as HTML, CSS, JavaScript, or other potentially harmful data. All information that comes from sources that are, for this purpose, considered untrusted, including user input and data stored in the database, is appropriately escaped to mitigate XSS attacks.

4 Policy and organisation

4.1 Data locality and physical security

The Wipster platform and services are developed in Wellington, New Zealand by WIP APP LIMITED. Wipster hosts its services and data on Microsoft's Azure cloud platform. Azure offers us a highly secure, robust, high-performance and scalable environment with a wide range of security controls. Azure is used by a large number of enterprise customers, including NBC, GE, 3M, BMW, and others.

Our platform is hosted in Azure's US West and US Central data centers, given the majority of our customer base operate in the United States.

Access to physical services hosting Wipster servers and data is tightly controlled and restricted by Microsoft, and subject to their strict security processes.

4.2 Security awareness and direction

Wipster has security processes and practices in place, guided by the Open Web Application Security Project (OWASP)[11]. Security policies and practices are continuously updated, based on technological and security-related developments.

Wipster software development is security-focussed and includes code reviews, change management, and security reviews of every new piece of code. Security implications of our code and infrastructure are reviewed frequently by the IT team. As part of our written "definition of done"— a checklist for new code that we must follow before it is merged into our main code base. We utilise automated tools such as Git Dependency Scanner to check all code for vulnerabilities before code is executed.

We maintain and frequently develop new technical documentation, for example around development processes and system administration. Most system administration tasks are automated and documented, to ensure a repeatable and well-tested process for provisioning or maintaining servers.

4.3 Team

Wipster's VP of Product and Engineering Robyn Haugh, has a Bachelor of Commerce (HONS) degree in Marketing and Strategy Management. She has 20 years of experience in product

[11] https://www.owasp.org/

development, channel and business management. Ultimate responsibility for Wipster's security policies and procedures, cloud infrastructure, and application code lie with Robyn.

System access for administrative purposes is only given to key Wipster employees, for the purpose of system maintenance, customer support, or other tasks that are necessary to ensure continued availability of the Wipster platform.

We have a technical team with a diverse skill base, covering backend and frontend development, along with product design. References are checked before hiring employees. Employment contracts contain confidentiality and intellectual property protection clauses.

Our development team is cross-functional: while every developer has expertise in certain areas, part of our development strategy is to ensure everyone can step in for someone else. This avoids being overly dependent on any individual.

Access to any sensitive information and our systems are based on individual access credentials, which are revoked when an employee leaves Wipster. Employees only have access to systems and information they need to perform their duties.

4.3.1 Third-party contractors

Wipster employs a third-party contractor in a Virtual CFO capacity. This individual does not have access to our Production systems or code, but has access to Wipster platform metadata to perform required analytics.

4.4 Staying up-to-date

Security policies, code, and documents are continuously updated and new security threats are assessed. We receive security and vulnerability notices for the systems and software we are using, such as the *Microsoft Technical Security Notifications* mailing list, the *Microsoft Security Newsletter*, and IT security news from other online sources.

4.5 Threat assessments

Threats and vulnerabilities are discussed frequently with the development team. Proper access control exists on all our systems, and production, staging, and development systems are strictly separated. We have an inventory of workstations and equipment, and a broad liability insurance for IT Companies.

Security risks of production systems, preview/staging systems and development systems (e.g. workstations, code repository, automated test systems, issue tracker and project management

systems) are assessed, and controls are in place to protect and isolate data. For instance, production data is never copied to development or staging systems and non-production systems have no access to production systems. Customer data is considered with the highest priority, followed by Wipster source code and the automated test environment.

Vulnerability assessments are frequently conducted by external third parties. If any vulnerabilities are reported, we have a policy in place that defines the timeframes for remediating the vulnerabilities. High vulnerabilities are resolved within 30 days. The root cause analysis of the vulnerabilities are assessed and issues are addressed so that we do not experience a recurrence of the issue.

5 Development practice

The Wipster platform and supporting systems are designed with security in mind. Every aspect of the design and implementation of our platform is viewed through IT security eyes, and in our development practice, we continuously ask questions such as:

- What security risks does this component present?
- How could we exploit this code?
- What could a potential attacker do with this information?
- Is this "secure by default?" In other words: if we introduce a bug or mis-configure this component, will it default to its strictest security settings?

Our development team uses the well-known source code management system git, hosted on a private repository on GitHub.com. Every new piece of functionality or bug fix is developed on a separate code branch, and only merged after a code review with at least one other member of the development team.

During a code review, other members of the development team ask questions and give suggestions, to maintain high standards of code quality. After all necessary changes to new code have been made and before the new code is merged into our code base, we go through our "definition of done" checklist, which consists of:

- Is error behaviour well-specified and tested?
- Are errors handled correctly and is no technical information leaked to the end-user?
- Are both code and automated test code reviewed, and is test coverage adequate?
- Are security impacts understood, and are informed decisions made by at least two team members for any code that could potentially have security impacts?
- If the code requires sensitive information or privileged access, is it clear why this is necessary, and are no more privileges granted than needed?
- Is all untrusted input, from the user, the database, or the file system, identified and appropriately escaped and sanitised?
- Is all output escaped appropriately, following OWASP guidelines?

6 System management

6.1 Security administration

Key Wipster employees have administrative access to our systems. System access, and access to the infrastructure management interface from Microsoft, is based on individual access credentials with multi-factor authentication that can be granted, revoked, or restricted when necessary.

Responsibilities and duties for IT personnel are stated in employment contracts, and management and the IT team are continuously communicating priorities and responsibilities.

VP of Product Robyn Haugh is ultimately responsible for IT architecture and security. Her role and responsibilities are documented, alongside those of other management team members.

Audit logs for administrative access are available on all systems that support it (e.g. Azure, Windows Event logs)

6.2 Logging, monitoring, and alerting

Systems and processes are in place to detect outages or abnormal system behaviour, using two primary mechanisms: Azure services and Pingdom. Pingdom is configured to send push notifications to our technical team in the event of system problems. Our SQL Server databases within Azure have automated alerts when database throughput unit (DTU) percentages are greater than 50% for 5 minutes and/or greater than 80% for 5 minutes. Web servers within Azure have similar alerts. Outages are reported via email and Slack to the Wipster technical team.

The Raygun error logging system collects application error events for our production servers. Logging dashboards provide quick overviews of the health of our systems and user activity, and detailed logs are available to diagnose faults. Raygun errors are deleted on a rolling basis as we hit our storage quota.

We also make use of the Seq logging platform, hosted on a virtual machine within our Azure environment, running on HTTPS with credentials required for access. This platform receives a variety of error and information-based output from the Wipster application. Events stored in Seq are deleted after 60 days.

Operational and incident response are handled by the CPO and senior developers.

6.3 Provisioning, testing, and automation

Provisioning of application servers is largely an automated process. We can provision a new server from scratch in less than twenty minutes. Our servers are configured to automatically scale up and down in response to demand.

Our production system is designed to avoid a single point of failure. At any point in time, at least two production servers are active to handle user activity. The dedicated database server has a dedicated slave on standby in a different data centre. All systems are hosted by Microsoft and utilise Azure's redundant and robust virtual server and network infrastructure.

Maintenance release windows generally occur during lower traffic times of the day. Our infrastructure allows us to deploy to production with virtually no impact to end users.

6.4 Change management

6.4.1 Application code changes

In almost all cases, new versions of the application will not change, delete, compromise, or make client data inaccessible. In rare, or unforeseen situations where access to data would be affected, we will consult with our customers before implementing the change.

Our development process requires all new code to comply with a "definition of done" checklist before it is merged into the code base. This includes code reviews, security impacts, adequate testing (both automated and functional) and documentation requirements.

New features are built in code branches prefixed with the name "feature/[id]-[name]", where [id] is the ID generated by our source code management system, GitHub, and [name] is a concise name of the given piece of work. Bugs found during testing are recorded in our issue tracker on GitHub. Post launch, any bugs are fixed in branches named with the prefix "bug/[id]-[name]".

Changes are merged into a release branch, which is used to deploy to production. Once changes are live to production and proved to be working to the required standard, the release branch is merged into our master branch. Master is considered to be the absolute source of truth for our production code.

Changes are tracked in GitHub. We have documented processes in place for code branches, merge requests, and other mechanisms to organise our source code changes, and new staff are trained to use these mechanisms and procedures. A summary of important changes is kept up-to-date in a Changes or Release Notes document that accompanies every new release.

6.4.2 Release planning

New versions of the application are tested on a staging system and any bugs are fixed on the feature-frozen release candidate code branch, before the release is deployed on our production platform.

Our development cycles vary, and we aim to release application updates regularly as they are ready. Staff within and outside of our technical team contribute to functionally testing changes. Planned releases are discussed within the technical team, with the CPO, and with other key stakeholders (CEO, customer support, etc.) to ensure opinions align before going live.

6.4.3 Deployment

Changes to production servers, including significant system changes, major new features, and bug fixes are tested locally on developer machines, and within a dedicated testing environment that does not share resources with production. Our “definition of done” list must be complete before we deploy changes to production.

6.5 Contingency and resilience

Wipster’s VP of Product, Robyn Haugh, is responsible for system and application security and the technical robustness of the Wipster platform. In her absence, our Technical Lead has delegated responsibility.

Information related to our business is stored in cloud business systems.

A documented process exists to set up a new production environment. Production systems, as well as source code management, automated testing, development and staging systems are all hosted off site in secure, redundant data centres managed by Microsoft and GitHub.

7 Technical risk mitigation

This section covers the Open Web Application Security Project's Top 10[12] list of most critical Web Application vulnerabilities and our mitigation strategies.

7.1 Injection

> *Injection flaws, such as SQL, OS, and LDAP Injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorisation.*

The Wipster platform does not construct queries directly from user input, but instead uses parameterised queries. This ensures malicious commands cannot be executed in our database servers. The Azure built-in *SQL Auditing & Threat Detection* capability is enabled for our database servers, which provides automated alerting when an attempted malicious attack is detected.

API functions in Wipster code that accept a range of parameters, such as key/value pairs, only accept those parameters they expect to see and strip out the rest.

7.2 Broken authentication and session management

> *Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.*

The Wipster platform uses a single set of strong authentication and session management controls (OAUTH2 Bearer tokens), as recommended by OWASP. Authentication occurs early in the web application middleware stack (see section 3.1) and meets most Level 1 and Level 2 and some Level 3 requirements of the *Application Security Verification Standard* sections V2 and V3[13].

See section 3.3.3 for more information on the implementation of our authentication mechanism and session handling.

[12] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[13] https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf

7.3 Cross-site scripting (XSS)

> *XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows a"ackers to execute scripts in the victim's browser which can hijack user sessions, deface websites, or redirect the user to malicious sites*

As described in section 3.4.4, we ensure that all untrusted data is properly escaped before it is displayed in client browsers. All fields are validated before accepting and storing them. Data from users is treated as untrusted.

7.4 Insecure object references

> *A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, a directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorised data*

Every resource that is accessed is subject to our authorisation code (see section 3.2), which is a uniform and well-understood framework that guards all our resources.

The web application API, made available over HTTPS to the web browser client application, uses symbolic names and does not expose any reference to namespaces or internal function names. API functions must be explicitly programmed as such before they are available: it is not possible to call arbitrary functions in the code.

7.5 Security misconfiguration

> *Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date*

Chapter 2 discusses our platform infrastructure and some of the measures we take to address security concerns. Furthermore, chapters 4, 5, and 6 discuss in detail our processes around change management and configuration.

Our servers are configured using automated, repeatable and reliable processes that take care to provision servers with up-to-date software, disable services we don't need, lock down access to the server, remove default accounts, perform automatic security updates, etc.

Our infrastructure uses strict access controls that isolate servers and services. For example, production, staging, and development environments are all strictly separate and use different, dedicated servers and credentials.

7.6 Sensitive data exposure

> *Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.*

Our systems do not store sensitive personally identifiable information (PII) such as credit card details, mailing address or similar. We do store company name and an optional user avatar. User accounts are identified by email address and password. We use *bcrypt*, cryptographically strong one-way encryption specifically designed for password encryption, to store password information. We never store or transmit passwords in plain text, and we take special care to prevent passwords, password crypts, or session ids from being displayed or logged to files. Section 3.3.2 explains in detail how we handle authentication and what measures we take to protect passwords. We discuss handling of session tokens in section 3.3.3.

Our databases (and their snapshots and backups) are encrypted at rest using Azure built-in encryption functionality, and all communication between our application servers and the database servers use TLS-encrypted channels. Communication with the database server is restricted to our Azure Cloud Services by default. Exceptions to this can only be granted explicitly via the database server firewall configuration (and are granted only for telecommuting purposes). The database backup retention period is 30 days, after which snapshots are deleted.

Our web and mobile applications can only be accessed via HTTPS, and the panel communicates with our private API over HTTPS.

The Wipster application supports non-authenticated use in the form of content reviewers. Authenticated users can share a piece of content either via a public share link, or via email address, (see section 3.3.4) to non-Wipster account holders. These users can only access resources to which they have a URI, and that link contains a cryptographically-strong token. All other access to application functionality requires both authentication and authorisation.

We discuss this in more detail in section 3.2.

7.7 Missing function level access control

> *Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorisation.*

We never rely on the JavaScript browser application for any authentication or authorisation. All API calls to the web application server are considered to be stateless and are subject to our authentication and authorisation mechanisms. Calls to our application must be authenticated unless a resource is explicitly declared as public, such as CSS files and certain background images. We use a uniform, consistent, and easily understood authorisation mechanism that guards our resources. Every request is verified and subject to authorisation checks. For more information about our authentication and authorisation mechanisms, see section 3.2.

7.8 Cross-site request forgery (CSRF)

> *A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.*

The Wipster web application is a single-page application based on Google's AngularJS upon which we have added CSRF protection. After successful login:

1. An anti-cross-site request forgery token is generated via in-built .NET anti-forgery methods, and stored in the browser's local storage.
2. A second token generated in the same way is placed as a cookie on the user's browser
3. The local storage token is automatically added to the HTTP headers by AngularJS when making requests to our server
4. The cookie token is sent by the browser with every request
5. In-built .NET methods validate that the tokens were the same as those issued
6. The submission of data to Wipster's servers is only allowed upon successful validation of the tokens

7.9 Using components with known vulnerabilities

> *Components, such as libraries, frameworks, and other software modules,*

> *almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defences and enable a range of possible attacks and impacts.*

In our system management practices, we ensure that we track vulnerability notices (e.g. from the *Microsoft Security Newsletter*) and new releases of frameworks we use. We use up-to-date server software, and we frequently update the frameworks we use to the latest stable versions.

Furthermore, we grant our software with the minimum privileges it needs to run properly. Our application server runs as a user that is especially created for this purpose, which has no access to the system beyond what it strictly needs to function. The application server does not run with full administrative privileges.

Lastly, our Cloud Services have the built-in Azure Antimalware feature enabled, which runs real-time assessments and sends alerts when threats are detected.

7.10 Unvalidated redirects and forwards

> *Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorised pages."*

The Wipster platform does not have functionality that allows users to set URLs, and we are not redirecting or forwarding users to external websites.

8 Disaster recovery plan

Wipster uses several mechanisms and processes to mitigate the impact of many types of events that could cause system outages.

8.1 Natural and accidental disasters

Because Wipster is hosted in Azure, we benefit from the robust infrastructure built by Microsoft. The data centres are equipped with climate control, fire detection/suppression, and power management systems (among others). Azure also protects against *forces majeures* such as

earthquakes, floods, and hurricanes via procedures tested on a regular basis, and because of their multi-data centre architecture.[14]

Our application server can be quickly restored via repeatable procedures. Our database server has a standby in another data centre that will failover automatically. We maintain encrypted database backups using third-party software from Cherry Safe[15], that are created automatically and backed up to a separate Azure data centre.

8.2 Disaster related to human actions

8.2.1 Loss of key staff

Ultimate responsibility for the Wipster technical platform rests with the VP of Product, Robyn Haugh. To reduce key-person risk, trusted senior developers have access to parts of our production infrastructure and related systems. Mechanisms are in place to revoke access when it is no longer required, or when an employee leaves.

8.2.2 Human error

Limited access to production systems combined with repeatable procedures to rebuild environments allows us to control risk without slowing down our small team.

8.2.3 Malicious internet-based attacks

Our application is designed to scale horizontally by adding more resources as demand ramps up. This can help during a distributed denial of service (DDOS) attack. Additionally, the Azure platform itself has a DDOS protection mesh on inbound, outbound, and cross-Azure region connectivity.[16]

8.3 Recovery plan

In case of a disaster that causes serious damage to Azure's physical infrastructure, they have DR procedures to recover operations. If a disaster affects all the geographically redundant data centres that host the Wipster platform, we can use our existing provisioning mechanisms to configure a production system in one of Microsoft's other data centres across the world.

Wipster has detailed disaster recovery plans and procedures to address narrow or wide outages and/or interruptions across the platform.

---

[14] Azure SOC information: https://www.microsoft.com/en-us/trustcenter/Compliance/SOC?downloadDocument=1&documentId=becfe3cc-5f7a-4a36-bd8a-351395fe38e7

[15] https://www.cherrysafe.com/

[16] https://docs.microsoft.com/en-us/azure/best-practices-network-security

9 Appendix

9.1 Wipster marketing website

Wipster makes use of the Squarespace platform to run our marketing website, available at https://wipster.io/. This system is completely separate from our application infrastructure on Azure, and is hosted entirely on Squarespace systems.

Key security features of this platform:

- Runs entirely on HTTPS
- Has a separate administration site for editing content
- Administration site requires credentialed access, with accounts created only for employees who need them for their job requirements. These accounts are created under *@wipster.io* email addresses.
- Information given by end users to Wipster via the Squarespace site is transported to Squarespace servers via HTTPS, and subsequently emailed by Squarespace to Wipster using SendGrid, which uses TLS to securely send email[17]

9.2 Azure compliance

Microsoft Azure holds a wide variety of certifications - more than any other cloud provider. For detail on compliance (e.g. SOC 1, SOC 2, ISO/IEC 27001), visit the Microsoft Trust Center at: https://azure.microsoft.com/en-us/support/trust-center/

9.3 Azure penetration testing

Microsoft have a defined process for conducting penetration testing against their cloud infrastructure: https://gallery.technet.microsoft.com/Cloud-Red-Teaming-b837392e

[17] https://sendgrid.com/blog/sendgrid-and-the-future-of-email-security/