Wagtail Security

As most of us know CMSes are vulnerable to malicious attacks due to the fact that they are built on open source frameworks. Wagtail release versions are numbered in the form A.B or A.B.C. The last one “C” is the patch release version number which states bugfix and security release. These releases will be 100% backwards-compatible with the previous patch release. Wagtail requires several common Django middleware modules to work and cover basic security. Wagtail provides its own middleware to cover these tasks. Some of the current Django security concerns that have been addressed in the past include:

Vulnerability CVE-2016-7401: An interaction between Google Analytics and Django's cookie parsing could allow an attacker to set arbitrary cookies leading to a bypass of CSRF protection.

Vulnerability CVE-2016-9013:When running tests with an Oracle database, Django creates a temporary database user. In older versions, if a password isn't manually specified in the database. A randomly generated password is now used for each test run.

Vulnerability CVE-2016-9014: Older versions of Django don't validate the Host header against settings. This makes them vulnerable to DNS binding attack.

Other aspects that are “covered” by Django security include:

* XSS Protection: Using Django templates protects you against the majority of XSS attacks. Django templates escape specific characters which are harmful to HTML.
* Cross site request forgery (CSRF) protection: CSRF protection checks every POST request for a secret. A malicious user cannot replay a form post to our website and have another user logged in. The malicious attacker would require the secret.

CsrfViewMiddleware will check that the HTTP “referer header” is set to a URL on the same origin (including subdomain and port).

* SQL Injection Protection: The use of arbitrary SQL code being executed on Django is prevented by using Django’s query sets. The SQL will be properly escaped by the underlying database driver.
* Click Jacking protection:  [X-Frame-Options middleware](https://docs.djangoproject.com/en/1.10/ref/middleware/#django.middleware.clickjacking.XFrameOptionsMiddleware) prevents a site from being rendered inside a frame (click jacking)
* HTTPS Security: HTTPS prevents malicious network users from sniffing around for authentication credentials.
* Session Security: The django.contrib.sessions framework prevents unauthorized access to subdomains.
* User uploaded content: Django sis customizable enough that it can allow for handlers to be disabled so that static files cannot be executed as code.