39 point HEROKU SECURITY CHECKLIST

from Expedited **WAF**

This checklist is intended to quickly give you a sense of how well your Heroku application fits best practices for the coding, configuration, and delivery

HEROKU PLATFORM SECURITY

Heroku as a Platform-As-A-Service starts with secure defaults and handles management of many of the underlying tasks like operating system updates, emergency SSL patches, and filtering of outward network attacks that you would otherwise need to handle yourself.

What you do need to consider:

1. On the latest Stack

Heroku Stacks are the underlying operating system image that your application is served from. Over time, like any OS, these get updated and you'll need to migrate your application

2. On the latest Buildpack

Buildpacks are sets of instructions for how your app code is deployed to the Stack. This includes the programming language and some dependencies. If you're using official buildpacks they will be updated as long as the Stack is still supported. Unofficial or Third-party buildpacks will need to be updated manually.

3. Latest Docker Updated?

If you're deploying Docker images to Heroku then you'll need to manually manage the underlying dependencies and updates in your Dockerfile.

4. Are you using Teams?

Heroku Teams are a way to logically group sets of applications under one set of billing, add-on and user groups. The main use cases are:

- If you are a consultant or agency: splitting app access cleanly between clients.
- If you are an organization deploying your own apps: cleanly splitting roles and responsibilities.

In either case, by decreasing the amount of access each individual team member has you reduce the potential for both unintentional and malicious actions.

5. Are you using Private Spaces?

Heroku Private spaces offer a network isolated group of apps and data services. This allows you additional

security and network options that aren't available to general applications. Some of the most critical:

- Data Stores like Postgres and Redis have additional security features
- You can specify at the ingress level what IP ranges can connect to the Private Space
- It is possible to load balance between private spaces located in different availability zones for added redundancy and resiliency. (With Expedited WAF or similar service)
- Connect to Amazon Web Services VPCs

6. Are you using Heroku Shield?

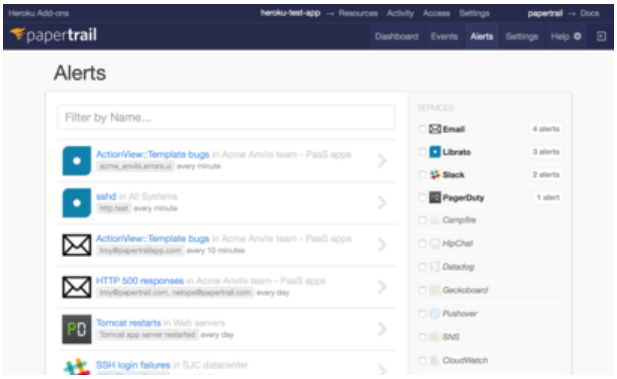
Heroku Shield is an additional set of features on top of the standard Heroku Private Spaces. It offers features to secure protected data classes like HIPAA, Financial or other sensitive data.

7. Enable Log Storage

By default, Heroku application logs are ephemeral. This presents a security problem as often it can be days or weeks until an issue is identified. Without detailed logs, it's impossible to track down what happened and when.

Coralogix

papertrail



8. Enforced SSL/TLS/HTTPS

Heroku by default supports older TLS 1.0 connections (for compatibility). Additionally, there is no option to force connections from HTTP -> HTTPS in the general Heroku stack.

If your application falls under GDPR, HIPAA, CCPA (other data security legislation) or if an external security audit will be performed, you will need to take additional steps.

Deciding which Heroku SSL/TLS option to use is more complicated than it sounds, so we made a guide on How To Choose What SSL/TLS/HTTPS option to use on Heroku.

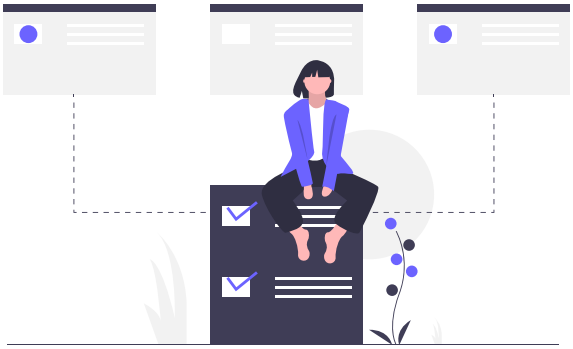
9. AWS Integrations

Heroku has an ephemeral file system, as a consequence, most applications that need to accept uploaded files save them to AWS S3.

S3 has become somewhat notorious was the service where people think data is stored securely up until the moment they realize that all of their sensitive files are open to anyone on the Internet.

APPLICATION LEVEL SECURITY

Your choice of a framework (if any), the dependencies you rely on and the way in which you configure and code all affect your application security.



10. Is your app framework up to date?

Underlying flaws in app frameworks can be patched (to a point) but regularly updating to newer versions helps prevent you from getting caught by architectural vulnerabilities.

11. Did you roll your own Crypto?

Cryptography is complicated and even incredibly minor differences in implementation can render the security of it useless. Rely as heavily as possible on open source or commercially supported authentication and encryption tools with widely understood non-proprietary algorithms.

12. Are you limiting the data you collect to the minimum?

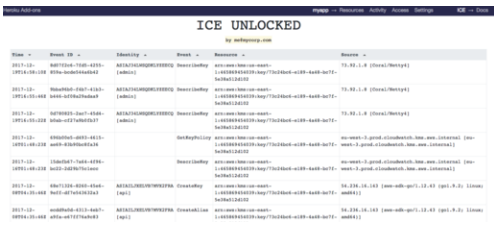
Data is best thought of as a toxic asset. If you don't have a clear and current use for a piece of user data then you should delete it (or ideally) not collect it in the first place.

13. Are you keeping credentials out of source control?

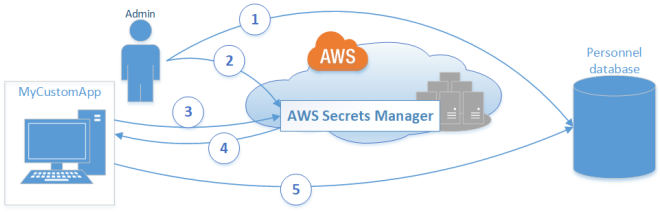
On Heroku, the best practice is to store credentials like API Keys in the Settings store of your individual Heroku application. You should also make sure that .env or other credential storage files used in development are excluded from source control (added to .gitignore).

If the credentials are especially sensitive, you can use an external Hardware Security Module service to store them.

ICE



AWS Secrets Manager



14. Establish Secure Coding Practices

For your framework, define what you consider to be the proper way of handling actions with the potential for introducing security vulnerabilities. Common actions are:

- - The adding of a new dependency
- - User Authentication (who is this?)
- - User Authorization (are they allowed to request this URL?)
- - Data storage
- - Escaping of HTML/CSS/JS

15. Add Security Questions to Code Reviews

As part of your code review template, add explicit questions to ask: "Is there anything in this that could potentially weaken the existing security of the application?"

16. Do you conduct code complexity and security analysis?

Static code analyzers can run as part of your CI/CD loop and identify potential issues before they are deployed to production.

17. Do you plan features with DDoS in mind?

Some features of your application will naturally take more processing resources than others. At the feature development stage for services like search, image processing or machine learning execution implement cutoff and contingency flags to give you manual control.

18. Are you applying security headers?

Modern browsers respect additional security response headers which can help prevent certain kinds of downgrade, phishing, or data extraction attacks.

19. Are you giving your users a 2FA option?

Two-factor authorization (of any kind) is superior to standard email/password authorization methods. Notably, this should be considered to be protection of your application _not_ of the individual user accounts.

20. Are you encrypting sensitive data at rest?

Heroku data stores offer encryption options which will automatically secure data at rest. If that's overkill for your application, consider framework-specific options to encrypt data in individual fields.

OWASP Top 10

The Open Web Application Security Project (OWASP) Top 10 is best thought of as a survey of web application developers. Updated annually, they are reporting on the types of attacks they are most frequently dealing with which helps you prioritize where your security development time is best spent.

The following ten questions line up with the latest OWASP Top 10 but are shifted slightly to make sense in the context of Heroku applications.

21. Are you blocking data injection attacks?

Examples would be SQL injection, XSS, or other malicious code.

22. Are you handling broken authentication?

Do you currently have a plan if you right now discovered that a bot was attempting a credential stuffing attack against your application?

23. Are you auditing sensitive data to make sure it isn't improperly exposed?

Reports, exports, user profiles and other aggregate views of data within your application can easily expose information that the user did not consent to or was unaware of. Collect only the data that you need and be extremely conservative in its use.

24. Are you catching parser errors?

XML, JSON or other data format parsing is an area that too often is ignored as safe (in a way that users pasting in HTML is not) but the consequences can be similar: improper data extraction, code injection, etc.

25. Are you implementing proper access controls?

In this context, access controls refer to continuously checking that users are authorized to access specific resources.

Ex: if a user can access their profile at: /users/123 (their user_id) can they read someone else's by manually entering /users/456

26. Are you auditing for security misconfigurations?

Most web frameworks have a "strict" mode which will help prevent categories of vulnerabilities. This may help enforce HTTPS-only connections or prevent string concatenation of SQL Queries (a classically dangerous action).

27. Are you insecurely deserializing data formats?

Transforming data structures from one form to another is a core aspect of nearly any web interaction (ex: form/url parameters to hashes or dictionaries).

But despite how common activity it is, the serialization/deserialization process can still be fraught with danger. Attackers can sneak in characters that break serialization in particular ways granting them further access outside the bounds of normal program execution.

28. Are you using components with known vulnerabilities?

No matter what web application framework you're using two things are guaranteed: new security issues are continually discovered and updates are necessary to patch and reduce the risk from vulnerable components and dependencies.

This has become more and more of an issue as frameworks on both the front and back end of applications have caught on. It's entirely possible that your application is dependent upon thousands of individual packages.

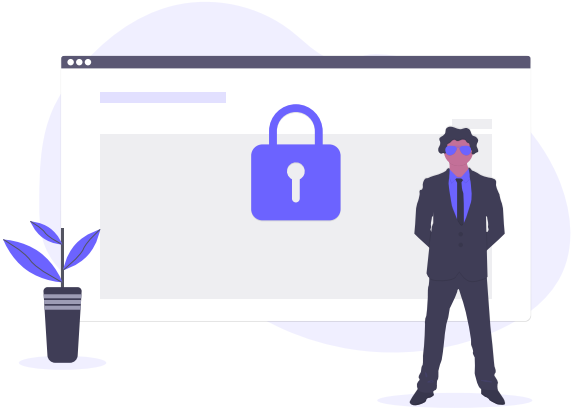
If you're using Github you can **configure automated security updates**.

29. Are you monitoring application usage?

We'd previously discussed logging (the historical collection of data on app usage) but the real-time collection is important as well. Sudden spikes in requests. Requests from previously unknown areas and requests that don't match normal patterns of behavior are all cause for investigation and potential action.

OUTSIDE SERVICES

Because your application is on the Internet, you're defacto relying upon a number of external services that all have a security impact. Locking them down is as important as preventing application attacks.



30. Are you protecting your DNS?

DNS underlies nearly all Internet services. If an attacker has control of your DNS they can trivially man-in-the-middle your traffic. Set up fake pages to collect user credentials and receive every email sent to your organization.

You should use 2FA on all DNS controlling accounts and (if available) set a registry lock on your domain.

31. Are you protecting your Email?

Email's role in operations (password resets, alerts, etc.) makes it a prime target and entry point to compromising your application. Make sure the 2fa is enabled and security is audited.

32. Are you auditing the services that your app depends upon?

Do you have processes in place to find out if your data storage, email, monitoring, or another 3rd party service is secure before you trust them with your user data? Do you have established communications channels to be alerted when they have security issues?

SECURITY PROCESSES

Tools alone can't make something secure (if you've ever forgotten to lock your door you know this to be true). Having clear processes for both day to day actions and plans for when things go awry is critical to maintaining application security.

33. Are you complying with data security regulations?

The full breadth of data security compliance is beyond the scope of this checklist. But in general: collect only the data you need, keep it only as long as necessary, provide people their own data if they ask, delete it if they ask and respond in a timely way.

34. Do you have a publicly published list of 3rd party services that may come in contact with user data?

This is necessary for most data security compliance. As an example see Paypal's 3rd party list https://www.paypal.com/uk/webapps/mpp/ua/third-parties-list

35. Do you have established security communications channels?

Having a security report of a vulnerability in your systems get caught up in customer support or misidentified as "not an issue" can make a security situation much worse. You should:

- Set up a security.txt file
  - View ours at https://expeditedsecurity.com/security.txt
  - More information on security.txt at https://securitytxt.org/
- Add security contacts to your site's contact page

36. Do you have an incident response plan?

It's much easier to plan for an issue before it happens than when you are in the midst of it. Some key points: do you have a means by which to contact all affected users? What level of a security vulnerability in your application would need to happen in order for you to turn off access? Do you have the means to investigate what occurred?

And perhaps most importantly, do you have the ability to reach all of your customers if some portion of your infrastructure is down?

Having a status page may help with the nuts and bolts of this

37. Do you have a password manager?

One of the largest behavioral differences between security professionals and ordinary developers is that the former all use password managers. They make it easy to use longer, more difficult to guess passwords that are unique to each service they use.

38. Do you have an onboarding / offboarding user checklist?

Dormant accounts that aren't removed can easily be a conduit for attacks. Data breaches from other sources can expose emails + reused passwords. Or malicious ex-employees may be able to take action against your service or users.

39. Do you have a disaster recovery plan?

Whether through malicious action (an attacker deleting your site) or through a catastrophic Internet event that takes out a portion of Heroku applications. Having a disaster recovery plan and contingencies in place ahead of time can make a world of difference.

Heroku has multiple options to distribute apps across regions or if you're using Expedited WAF you can configure cross-cloud systems in failover or load-balanced configurations.

Heroku Private Spaces

Expedited WAF Network Plan

ABOUT

A team of dedicated security professionals

Application and Security Consulting

We got our start building custom web applications for the US Navy, 3M Health Information Services and other high security environments with significant compliance requirements.

To this day our team is still primarily drawn from the veteran community.

Expedited SSL

In 2014 we launched Expedited SSL - a service to automate the installation, maintenance and renewal of SSL/TLS certificates on cloud platforms.

IP Investigator

In 2016 we released a public API exposing our network investigation tools, letting developers build threat intelligence scoring into their own applications.

Expedited WAF

In 2019 we relased Expedited WAF - a preconfigured and tuned Web Application Firewall that can safely be dropped in front of any web app to instantly improve its security profile.

Today

We're continuing on our mission to deliver security products and services that developers can readily use to protect user data, secure their SAAS and feel confident in their security posture.

ExpeditedWAF

BOOK A DEMO