

JUN 1, 2020

Web Developer Security Checklist V2



Developing secure, robust web applications in the cloud is **hard**, very hard. If you think it is easy, you are either a higher form of life or you have a painful awakening ahead of you.

If you have drunk the MVP cool-aid and believe that you can create a product in one month that is both valuable and secure — think twice before you launch your “proto-product”.

After you review the checklist below, acknowledge that you are skipping many of these critical security issues. At the very minimum, be *honest* with your potential users and let them know that you don’t have a complete product yet and are offering a prototype without full security.

This checklist is simple, and by no means complete. I’ve been developing secure web applications for over 14 years and this list contains some of the more important issues that I’ve painfully learned over this period. I hope you will consider them seriously when creating a web application.

This is version 2 of the checklist. It has been re-organized from Version 1 and has a few new items by public demand (Thank y While I try to keep the list tight and focused, please comment if you have an item that you think I should add to the list.

- Credentials and Secrets
- Authentication
- Database
- Apps
- APIs
- Network Traffic
- Cloud Configuration
- Infrastructure
- Denial of Service Protection
- Hack Yourself
- Incident Response

Credentials and Secrets

- ☐ Store and distribute secrets using a key store designed for the purpose. Don't hard code secrets in your applications and definitely don't store in GitHub!. For CMS fans, don't store your credentials in a file in the document directory.
- ☐ Use a team-based password manager such as 1Password for all service passwords and credentials. NEVER email passwords or credentials to team members.
- ☐ Use multi-factor authentication for all your logins to service providers.

Authentication

- ☐ Ensure all passwords are hashed using appropriate crypto such as bcrypt. Never write your own crypto and correctly initialize crypto with good random data. Consider using an authentication service like Auth0 or AWS Cognito.
- ☐ Use best-practices and proven components for login, forgot password and other password reset. Don't invent your own — it is hard to get it right in all scenarios.
- ☐ Implement simple but adequate password rules that encourage users to have long, random passwords.
- ☐ Never, EVER have any undocumented and unpublicized means of access to the device including back-door accounts (like "field-service").
- ☐ Run applications and containers with minimal privilege and never as root (Note: Docker runs apps as root by default).

Database

- ☐ Don't store sensitive data unless you truly need it. This means email addresses, personally identifying information and other personal information in general. Treat sensitive data like radioactive waste — i.e. there is an real, large and ongoing cost to securing it, and one day it can hurt you.
- ☐ Keep a complete list of all the places you store sensitive information: databases, file systems, Dropbox, GitHub, Vault, Office docs and even the paper folder. This is useful to manage, required by GDPR and essential if hacked. You need to be able to locate all sensitive information.
- ☐ If subject to GDPR, make sure you really understand the requirements and design it in from the start. For some, it will represent a major change in design and thinking. See Privacy Cheatsheet and Intro to GDPR.
- ☐ Use encryption for data identifying users and sensitive data like access tokens, email addresses or billing details if possible (this will restrict queries to exact match lookups).
- ☐ If your database supports low cost encryption at rest (like AWS Aurora), then enable that to secure data on disk. Make sure all backups are stored encrypted as well.

- ☐ Fully prevent SQL injection by only using SQL prepared statements. For example: if using NPM, don't use npm-mysql, use npm-mysql2 which supports prepared statements.

Apps

- ☐ Secure development systems with equal vigilance to what you use for production systems. Build the software from secured, isolated development systems.
- ☐ Ensure that all components of your software are scanned for vulnerabilities for every version pushed to production. This means O/S, libraries and packages. This should be automated into the CI-CD process.
- ☐ Do client-side input validation for quick user feedback, but never trust it. Always validate and encode user input before displaying. Here is a useful checklist [Client Side Checklist](#).
- ☐ Validate every last bit of user input using white lists on the server. Consider generating validation code from API specifications using a tool like Swagger, it is more reliable than hand-generated code.
- ☐ Never directly inject user content into responses. Never use untrusted user input in SQL statements or other server-side logic.
- ☐ Use centralized logging for all apps, servers and services. You should never need SSH to access or retrieve logs. On AWS, consider CloudWatch with the [SenseLogs Viewer](#).
- ☐ Log with sufficient detail to diagnose all operational and security issues and NEVER log sensitive or personal information. Consider creating logs in JSON with high cardinality fields rather than flat text lines.
- ☐ Don't emit revealing error details or stack traces to users and don't deploy your apps to production with DEBUG enabled.

APIs

- ☐ Ensure that users are fully authenticated and authorized appropriately when using your APIs.
- ☐ Ensure that no resources are enumerable in your public APIs. For IDs, consider using RFC 4122 compliant UUIDs instead of integers. For node, see [NPM uuid](#).
- ☐ Use canary checks in APIs to detect illegal or abnormal requests that indicate attacks.

Network Traffic

- ☐ Segment your network and protect sensitive services. Use firewalls, virtual private networks and cloud Security Groups to restrict and control inbound and outbound traffic to/from appropriate destinations. AWS and CloudFlare both have excellent offerings.
- ☐ Use TLS for the entire site, not just login forms and responses. Never use TLS for just the login form. Transitionally, use the strict-transport-security header to force HTTPS on all requests.
- ☐ Cookies must be httpOnly and secure and be scoped by path and domain.
- ☐ Use CSP without allowing unsafe-* backdoors. It is a pain to configure, but worthwhile. Use CSP Subresource Integrity for CDN content.
- ☐ Use X-Frame-Option, X-XSS-Protection headers in client responses. Use <https://observatory.mozilla.org> to score your site.
- ☐ Use HSTS responses to force TLS only access. Redirect all HTTP request to HTTPS on the server as backup.
- ☐ Use CSRF tokens in all forms and use the new SameSite Cookie response header which fixes CSRF once and for all newer browsers.
- ☐ Remove other identifying headers that can make a hackers job easier of identifying your stack and software versions.
- ☐ Don't use GET requests with sensitive data or tokens in the URL as these will be logged on servers and proxies.

- ☐ Ensure all services have minimum ports open. While security through obscurity is no protection, using non-standard ports will make it a little bit harder for attackers.
- ☐ Host backend database and services on private VPCs that are not visible on any public network. Be very careful when configuring AWS security groups and peering VPCs which can inadvertently make services visible to the public.
- ☐ Create test and staging resources in a separate AWS account to that used by production resources.
- ☐ Isolate logical services in separate VPCs and peer VPCs to provide inter-service communication.
- ☐ Ensure all services only accept data from a minimal set of IP addresses.
- ☐ Restrict outgoing IP and port traffic to minimize APTs and “botification”.
- ☐ Always use AWS IAM roles and not root credentials.
- ☐ Use minimal access privilege for all ops and developer staff.
- ☐ Regularly rotate passwords and access keys according to a schedule.

Infrastructure

- ☐ Ensure you can do upgrades without downtime. Ensure you can quickly update software in a fully automated manner.
- ☐ Create all infrastructure using a tool such as Terraform, and not via the cloud console. Infrastructure should be defined as “code” and be able to be recreated at the push of a button. Have zero tolerance for any resource created in the cloud by hand – Terraform can then audit your configuration.
- ☐ Don’t SSH into services except for one-off diagnosis. Using SSH regularly, typically means you have not automated an important task.
- ☐ Don’t keep port 22 open on any AWS service groups on a permanent basis. If you must use SSH, only use public key authentication and not passwords.
- ☐ Create immutable hosts instead of long-lived servers that you patch and upgrade. (See [Immutable Infrastructure Can Be More Secure](#)).
- ☐ If not using Immutable Infrastructure (bad), ensure you have an automated system to patch and update all servers and regularly update your AMIs and rotate your servers to prevent long-lived APTs.
- ☐ Power off unused services and servers. The most secure server is one that is powered down. Schedule dev servers to be powered down after hours when not required.
- ☐ Use an Intrusion Detection System to minimize APTs.

Denial of Service Protection

- ☐ Make sure that DOS attacks on your APIs won’t cripple your site. At a minimum, have rate limiters on your slower API paths and authentication related APIs like login and token generation routines. Consider CAPTCHA on front-end APIs to protect back-end services against DOS.
- ☐ Enforce sanity limits on the size and structure of user submitted data and requests.
- ☐ Perform Chaos testing to determine how your service behaves under stress.
- ☐ Consider using Distributed Denial of Service (DDOS) mitigation via a global caching proxy service like CloudFlare. This can be turned on if you suffer a DDOS attack and otherwise function as your DNS lookup.

Hack Yourself

- ☐ Audit your design and implementation.





☐ Proactively test your app beyond normal use. Consider the OWASP test checklist to guide your test hacking.

Incident Response


- ☐ Train staff (especially senior staff) as to the dangers and techniques used in security social engineering.
- ☐ Have a threat model that describes what you are defending against. It should list and prioritize the possible threats and actors.
- ☐ Setup a standard email account and web page dedicated for users to report security issues (security@example.com and /security).
- ☐ Have a practiced security incident plan. One day, you will need it.


Security is a Journey

Most of all, remember that security is a journey and cannot be "baked-in" to the product just before shipping. I hope this checklist will prompt you through your entire development lifecycle to improve the security of your services.

Version 1 of this checklist can be found at [Web Developer Security Checklist V1](#).

Learn More About SenseLogs



SenseLogs 

MOB@EMOBRIEN.COM






DOC

HELP

Sample App Errors

Jan 7, 0:56

type == "error"



SAVE

Timestamp	Type	Message	Name	Email	Ip_address	User_agent
Jan 5, 17:00:00	error	lectus pellentesque eget nunc donec quis	Christopher Deamer	cdeamerh@cnet.com	14.100.10.34	Mozilla/5.0 (Macir
Jan 6, 2:43:20	error	amet consectetur adipiscing elit proin risus praes...	Rhys Warters	rwarterso@yandex.ru	52.152.140.152	Mozilla/5.0 (Macir
Jan 6, 20:46:40	error	tincidunt in leo maecenas pulvinar lobortis est pha...	Milicent Dallicott	mdallicott11@seesaa.net	233.162.0.252	Mozilla/5.0 (Macir
Jan 7, 0:56:40	error	pede ullamcorper augue a suscipit nulla elit ac	Tracee Bamfield	tbamfield14@usnews.com	240.212.113.98	Mozilla/5.0 (X11; t

email: tbamfield14@usnews.com

id: 41

ip_address: 240.212.113.98

message: pede ullamcorper augue a suscipit nulla elit ac

name: Tracee Bamfield

request_id: 3ea5239b3d9cc774aeefca458d9a6703

timestamp: Jan 7, 0:56:40

type: error

user_agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:17.0) Gecko/20100101 Firefox/17.0.6

user_id: 313a2fe9471d9d2583c9cfd1d77e8f61f

Jan 7, 2:20:00	error	integer pede justo lacinia eget tincidunt eget temp...	Montague Wickling	mwickling15@usda.gov	76.85.220.151	Mozilla/5.0 (X11; l
Jan 7, 6:30:00	error	sit amet erat nulla tempus vivamus in felis eu sapi...	Ham Carvilla	hcarvilla18@cbc.ca	116.224.79.102	Mozilla/5.0 (Macir
Jan 8, 3:20:00	error	iaculis congue vivamus metus arcu adipiscing mol...	Salaith Brixham	sbrixham1n@eepurl.com	20.52.168.74	Mozilla/5.0 (X11; l
Jan 9, 1:33:20	error	cubilia curae donec pharetra magna vestibulum ali...	Philomena Bourdas	pbourdas23@go.com	158.228.217.158	Mozilla/5.0 (Windi
Jan 9, 11:16:40	error	primis in faucibus orci luctus et	Nicolas Gillis	ngillis2a@comsenz.com	165.236.216.241	Mozilla/5.0 (Windi
Jan 11, 2:10:00	error	est congue elementum in hac habitasse	Annemarie Goudy	agoudy32@dedecms.com	87.48.134.162	Mozilla/5.0 (iPhon
Jan 11, 14:40:00	error	lectus pellentesque eget nunc donec quis orci eget...	Humberto Genese	hgenese3b@geocities.com	251.192.67.113	Mozilla/5.0 (Macir
Jan 12, 4:33:20	error	cursus urna ut tellus nulla ut erat id mauris vulputa...	Haily Marden	hmarden3l@chronoengine....	127.45.218.62	Mozilla/5.0 (Windi
Jan 13, 12:30:00	error	et ultrices posuere cubilia curae nulla dapibus dolo...	Armin Jessop	ajessop48@drupal.org	246.176.38.231	Mozilla/5.0 (iPad; t
Jan 14, 20:26:40	error	semper interdum mauris ullamcorper purus sit am...	Saxe Crickmer	scrickmer4v@walmart.com	195.177.2.191	Mozilla/5.0 (Windi

While developing cloud services at SenseDeep, we wanted to use CloudWatch as the foundation for our logging infrastructure, but we needed a better, simple log viewer that supported fast smooth scrolling and better log data presentation.

So we created SenseLogs, an AWS CloudWatch Log solution that runs blazingly fast, 100% in your browser. It transparently downloads and stores log events in your browser application cache for immediate and later viewing. It offers smooth scrolling, live tail and powerful structured queries. It understands structured log data for easy presentation and queries.

Try it for free at: <https://app.senselogs.io> or learn more at: <https://www.sensedeepp.com/senselogs>.

Please let us know what you think, we thrive on feedback: dev@sensedeepp.com.

Jun 1, 2020 • AWS, Checklist, Dev, Ops, Security, All Posts

Be notified for new stories and tips

<input type="text" value="Email address ..."/>	<input type="button" value="Subscribe"/>
--	--