Aha-Secret

Owner: AHA-OIDA
Reviewer: whotwagner
Contributors:
Date Generated: Sun Aug 31 2025

Executive Summary

High level system description

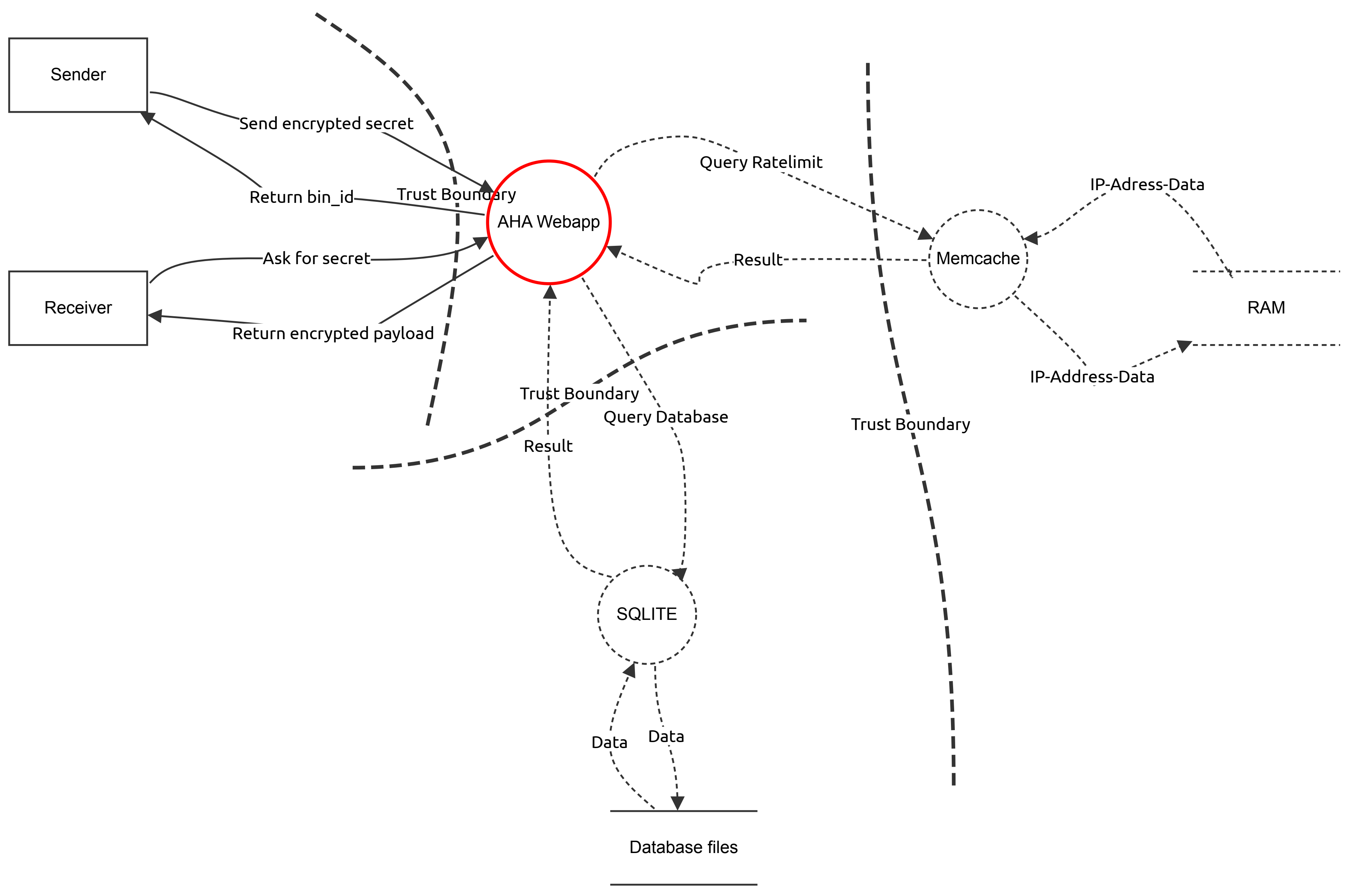
aha-secret allows you to store a secret message encrypted. Only the generated link could decrypt the message again. The message is encrypted by your browser and decrypted by the browser of the person who is allowed to read it. The first time someone clicks on the link, the message is automatically deleted from the server.

Summary

Total Threats	34
Total Mitigated	32
Total Not Applicable	1
Total Open	1
Open / Critical Severity	0
Open / High Severity	1
Open / Medium Severity	0
Open / Low Severity	0

STRIDE High Level

High level STRIDE-Diagram



STRIDE High Level

Send encrypted secret (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
15	Attacker fills up disk by sending data masses	Denial of service	Medium	Mitigated		Attacker posts very large amount of data in order to fill up the disk	* Ratelimit via Rack::Attack * Low Sizelimit for payload
16	Attacker fills up disk by sending many requests	Denial of service	Medium	Mitigated		Attacker sends many requests to fill up disk or exhaust other resources	Ratelimit via Rack::Attack
20	Attacker sends XSS-Payload to Reciever	Tampering	Medium	Mitigated		An attacker could send xss-payload to reciever to execute malicious javascript in the webrowser of the reciever	* Using Content Security Policy using Rack::Protection * Textarea limits possible payloads for xss
21	Attacker injects SQL-commands	Tampering	Medium	Mitigated		Attacker injects sql commands	* Using well known ActiveRecord-gem that uses safe methods to interact with the database * Every record has an encrypted payload and the keys are never on the server
41	An eavesdropper can read the secret payload	Information disclosure	High	Mitigated		An eavesdropper can read the secret payload	* The payload is encrypted with AES256-GCM and the secret is never sent to the server * The webapp(crypto-api.js) only works with HTTPS or on local-installations
68	Encryption can be broken because of re-used Initial-Vectors	Information disclosure	Critical	Mitigated		Encryption can be broken if the same initial vector is used multiple times	Always use strong and cryptographic safe algorithm to generate random iv's
70	An attacker could send a victim malicious java-script-code	Tampering	Medium	N/A		An attacker could inject javascript to trick the victim to click on a link or show the victim malicious content(porn, etc...)	Strong CSP-Policy and CORS mitigates XSS-Attacks

Return bin_id (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
38	An eavesdropper could intercept the traffic and read out data	Information disclosure	Low	Mitigated		An eavesdropper could intercept the traffic and read out data. For example the bin_id.	The javascript crypto-library does only work if the backend runs on localhost or uses HTTPS
43	Adversary-in-the-Middle could tamper data	Tampering	Medium	Mitigated		Adversary-in-the-Middle could tamper data to produce xss or other malicious payloads	* HTTPS * HSTS
47	Adversary-in-the-Middle could tamper data to disrupt the service	Denial of service	Medium	Mitigated		Adversary-in-the-Middle could tamper data to disrupt the service	* HTTPS * HSTS

Return encrypted payload (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
42	An eavesdropper can read the secret payload	Information disclosure	High	Mitigated		An eavesdropper can read the secret payload	* The payload is encrypted with AES256-GCM and the secret is never sent to the server * The webapp(crypto-api.js) only works with HTTPS or on local-installations
44	Adversary-in-the-Middle could tamper data	Tampering	Medium	Mitigated		Adversary-in-the-Middle could tamper data to produce xss or other malicious payloads	* HTTPS * HSTS

Number	Title	Type	Severity	Status	Score	Description	Mitigations
48	Adversary-in-the-Middle could tamper data to disrupt the service	Denial of service	Medium	Mitigated		Adversary-in-the-Middle could tamper data to disrupt the service	* HTTPS * HSTS

Ask for secret (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
25	Enumerate possible payload-ids	Information disclosure	Medium	Mitigated		An attacker could try to enumerate possible id numbers of payload in order to delete secrets of other people.	Using secure token of the ActiveRecord
40	An eavesdropper could intercept the traffic and read out data	Tampering	Low	Mitigated		An eavesdropper could intercept the traffic and read out data. For example the bin_id.	The javascript crypto-library does only work if the backend runs on localhost or uses HTTPS
49	Attacker disrupts service with flood of requests	Denial of service	Medium	Mitigated		Attacker disrupts service with flood of requests	Ratelimit via Rack::Attack
56	Reciever accidentially reveals secret and deletes it	Tampering	Medium	Mitigated		Reciever accidentially reveals secret and deletes it. For example, a sender creates a secret and clicks on the link instead of copy it.	Just clicking on the link does not retrieve the secret. Only a PATCH-Operation(/reveal) will fetch the secret and deletes it from the server

Database files (Store)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
58	Dataleak	Information disclosure	High	Mitigated		The database-files could be accessed by unauthorized entities. For example by: * attackers hacked the system * administrators steal the data * old harddisk * aso...	* Data is encrypted on client side * Key is never passed on the server * Strong encryption algorithm is used on the client side * Data is decrypted on client side
59	Attacker on server-side could manipulate data of any user	Tampering	High	Mitigated		Attacker on server-side could manipulate data of any user	* Data is encrypted on client side * Key is never passed on the server * Strong encryption algorithm is used on the client side * Data is decrypted on client side
65	Decryption link leaked and an attacker got the secret	Information disclosure	High	Mitigated		AHA-Secret is not responsible for how the decryption-link is shared. If the link is somehow disclosed by the user and an unauthorized entity fetches the key, the secret will be compromised	Additional Password prevents this

AHA Webapp (Process)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
10	Supply-Chain-Attack: Backdoored dependencies	Elevation of privilege	High	Mitigated		A backdoored dependency could compromise the server. An attacker could replace the javascript files to extract the secrets on the client side.	* Reduce dependencies: no nodejs
11	Vulnerable dependency allows code execution	Elevation of privilege	High	Mitigated		A vulnerability in any dependency or 3rd-party software opens a remote code execution vulnerability	* Minimum features -> less code -> keep attack vector small * Dependabot in Github checks for updates

Number	Title	Type	Severity	Status	Score	Description	Mitigations
55	Attacker on server-side could manipulate data of any user	Spoofing	High	Mitigated		Attacker on server-side could manipulate data of any user	* Data is encrypted on client side * Key is never passed on the server * Strong encryption algorithm is used on the client side * Data is decrypted on client side
51	Attacker on server-side could manipulate javascript to exfiltrate secrets	Tampering	High	Open		Attacker on server-side could manipulate javascript to exfiltrate secrets on the client-side.	
52	Decryption link leaked and an attacker got the secret	Repudiation	High	Mitigated		AHA-Secret is not responsible for how the decryption-link is shared. If the link is somehow disclosed by the user and an unauthorized entity fetches the secret, it must be detected.	* One-Time-Secrets only! The reciever will notice that the link does not work anymore. * Access-Logs * Custom passwords could prevent to read the secret with the link only
53	Dataleak	Information disclosure	High	Mitigated		The data could leak due to various reasons: * attackers hacked the system * administrators steal the data * old harddisk * aso...	* Data is encrypted on client side * Key is never passed on the server * Strong encryption algorithm is used on the client side * Data is decrypted on client side
54	DoS because of floods of requests	Denial of service	Medium	Mitigated		DoS because of floods of requests	Requst Ratelimit based on IP using Rack::Attack
62	Server-side attacker could guess encryption-key	Information disclosure	High	Mitigated		An attacker is able to get the database and tries to break the encryption	Client-side autogenerated cryptographic secure random keys prevent such attacks against AES256
64	Decryption link leaked and an attacker gets the secret	Information disclosure	High	Mitigated		AHA-Secret is not responsible for how the decryption-link is shared. If the link is somehow disclosed by the user and an unauthorized entity fetches the key, the secret will be compromised	Additional custom password prevents it
73	Weak custom password leads to info-leak	Information disclosure	Critical	Mitigated		The sender uses a weak custom password that can be bruteforced on the server. Such a scenario could explose the secrets on the server side	Use custom passwords only as additional passwords. The secret is still mainly encrypted with a strong random-key

Sender (Actor)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
28	An attacker forces an end user to execute unwanted actions on the webapp(CSRF)	Spoofing	Low	Mitigated		An attacker forces an end user to execute unwanted actions on the webapp(CSRF)	Using csrf-token and CORS via Rack::Protection
32	Attacker denies sending payload	Repudiation	Low	Mitigated		Attacker denies sending malicious payloads	Access-Logs

Receiver (Actor)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
29	An attacker forces an end user to execute unwanted actions on the webapp(CSRF)	Spoofing	Low	Mitigated		An attacker forces an end user to execute unwanted actions on the webapp(CSRF)	Using csrf-token and CORS via Rack::Protection
33	Attacker denies sending payloads	Repudiation	Low	Mitigated		Attacker denies sending malicious payloads	Access-Logs