



OUT OF BOUNDS

ANALISI SULLE VULNERABILITÀ
PROGETTO CAPTCHA
Versione 1.0.0

Responsabile	Michele Cazzaro
Redattore	Michele Cazzaro Alberto Matterazzo Valentina Caputo Simone Bisortole
Verificatore	Alberto Matterazzo
Uso	Esterno
Destinatari	Out of Bounds prof. Tullio Vardanega Prof. Riccardo Cardin Zucchetti S.p.A.

CONTATTI

sweoutofbounds@gmail.com

REPOSITORIES

[orgs/SWE-OutOfBounds/repositories](https://github.com/SWE-OutOfBounds/repositories)

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
1.0.0	2023/05/28	Michele Cazzaro	Responsabile	Approvazione e rilascio del documento.
0.1.0	2023/05/27	Alberto Matterazzo	Verificatore	Verifica generale del documento.
0.0.7	2023/05/24	Valentina Caputo, Alberto Matterazzo	Programmatore, Verificatore	Stesura \$5,\$6,\$7 e verifica.
0.0.6	2023/05/23	Valentina Caputo, Alberto Matterazzo	Programmatore, Verificatore	Stesura \$4 e verifica.
0.0.5	2023/05/22	Simone Bisortole, Alberto Matterazzo	Programmatore, Verificatore	Stesura \$3 e verifica.
0.0.4	2023/05/19	Simone Bisortole, Alberto Matterazzo	Programmatore, Verificatore	Stesura \$2 e verifica.
0.0.3	2023/05/18	Michele Cazzaro, Alberto Matterazzo	Responsabile, Verificatore	Stesura \$1 e verifica.
0.0.2	2023/05/18	Michele Cazzaro	Responsabile	Creazione scheletro documento.
0.0.1	2023/05/13	Valentina Caputo	Amministratore	Creazione documento.

Tabella 1: Registro delle modifiche

Indice

1	Introduzione	3
1.1	Obiettivo dell'analisi	3
1.2	Contesto e importanza di <i>web-app</i> e libreria	3
1.2.1	Contesto della libreria <i>CAPTCHA_G</i>	3
1.2.2	Importanza della libreria <i>CAPTCHA_G</i>	4
1.2.3	Contesto della <i>web-app</i> dimostrativa	4
1.2.4	Importanza della <i>web-app</i> dimostrativa	4
1.3	Struttura del documento di analisi	4
2	Descrizione della libreria <i>clock-captcha</i>	6
2.1	Scopo e funzionamento della libreria <i>clock-captcha</i>	6
2.2	Tecnologie utilizzate	6
3	Metodologia di analisi	7
3.1	Approccio seguito per identificare le vulnerabilità	7
3.2	Strumenti utilizzati durante l'analisi	7
4	Analisi delle vulnerabilità	9
4.1	Identificazione delle possibili vulnerabilità	9
4.2	Descrizione delle vulnerabilità individuate	12
5	Conclusioni	15
6	Raccomandazioni	16
6.1	Suggerimenti per mitigare le vulnerabilità identificate	16
6.2	Indicazioni per una corretta configurazione della libreria <i>CAPTCHA_G</i>	16
7	Riferimenti	17
7.1	Fonti utilizzate per l'analisi	17
7.2	Documentazione della libreria <i>CAPTCHA</i> e dell'applicazione	17

1 Introduzione

1.1 Obiettivo dell'analisi

L'obiettivo dell'analisi divulgata in questo documento è identificare e valutare le potenziali vulnerabilità e rischi associati all'implementazione e all'utilizzo della libreria *CAPTCHA_G* sviluppata dal gruppo *Out of Bounds* e della sua *web-app* dimostrativa. L'analisi delle vulnerabilità ha diversi scopi principali:

- **Identificazione delle vulnerabilità:** L'analisi mira a individuare le potenziali vulnerabilità presenti nella libreria *CAPTCHA_G* e nella *web-app* dimostrativa. Questo può includere vulnerabilità di codice, errori di configurazione, debolezze di sicurezza e altre falle che potrebbero essere sfruttate da attaccanti malevoli,
- **Valutazione dell'impatto potenziale:** L'analisi delle vulnerabilità cerca di comprendere l'impatto che ciascuna vulnerabilità potrebbe avere sulla libreria *CAPTCHA_G* e sulla *web-app* dimostrativa. Si valuta il livello di rischio associato a ciascuna vulnerabilità considerando le conseguenze che potrebbero verificarsi in termini di compromissione della sicurezza, perdita di dati sensibili o violazione dell'integrità dell'applicazione,
- **Fornire raccomandazioni di sicurezza:** L'analisi delle vulnerabilità fornisce raccomandazioni per mitigare le vulnerabilità identificate e migliorare la sicurezza della libreria *CAPTCHA_G* e della *web-app* dimostrativa. Ciò può includere suggerimenti per correggere il codice, implementare contromisure di sicurezza, migliorare la gestione degli accessi e delle sessioni o adottare pratiche di sviluppo sicure,
- **Sensibilizzazione e consapevolezza:** L'analisi delle vulnerabilità aiuta a sensibilizzare lo sviluppatore, il team di sicurezza e gli utenti della libreria *CAPTCHA_G* e della *web-app* dimostrativa riguardo alle possibili minacce e vulnerabilità. Aumentare la consapevolezza sulla sicurezza può incoraggiare la corretta implementazione delle misure di protezione e favorire un atteggiamento proattivo verso la sicurezza delle applicazioni.

1.2 Contesto e importanza di *web-app* e libreria

Il contesto e l'importanza della libreria *CAPTCHA_G* e della *web-app* dimostrativa possono variare a seconda della situazione specifica in cui vengono utilizzate. Tuttavia, di seguito sono forniti alcuni punti generali che illustrano il contesto e l'importanza di tali componenti.

1.2.1 Contesto della libreria *CAPTCHA_G*

- La libreria *CAPTCHA_G* è progettata per fornire un meccanismo di sicurezza per verificare che un utente sia umano e non un software automatizzato o un *bot*,
- Viene comunemente utilizzata in applicazioni *web* per prevenire attacchi automatizzati come lo *spam*, l'iniezione di contenuti indesiderati o la manipolazione dei dati,
- Le librerie *CAPTCHA_G* sono utilizzate in una vasta gamma di applicazioni, inclusi *form* di registrazione, pagine di accesso, funzioni di recupero password e altre interazioni utente-autenticazione.

1.2.2 Importanza della libreria *CAPTCHA_G*

- La libreria *CAPTCHA_G* svolge un ruolo cruciale nella protezione delle applicazioni *web* da attacchi automatizzati e *bot* malevoli,
- Senza un meccanismo di verifica efficace come il *CAPTCHA_G* le applicazioni possono essere vulnerabili ad attacchi di forza bruta, *spamming* automatizzato e altre forme di abusi,
- Una libreria *CAPTCHA_G* robusta e ben implementata può contribuire a migliorare la sicurezza generale dell'applicazione e proteggere i dati degli utenti.

1.2.3 Contesto della *web-app* dimostrativa

- La *web-app* dimostrativa è un'applicazione di esempio o un prototipo che viene fornito insieme alla libreria *CAPTCHA_G* per mostrare il suo funzionamento e le sue funzionalità,
- Serve come strumento per dimostrare l'uso corretto della libreria *CAPTCHA_G* e come esempio pratico per gli sviluppatori che desiderano integrarla nelle proprie applicazioni,
- Può essere utilizzata per mostrare l'interazione tra la libreria *CAPTCHA_G* e l'applicazione ospitante e fornire un esempio di implementazione sicura.

1.2.4 Importanza della *web-app* dimostrativa

- La *web-app* dimostrativa gioca un ruolo importante nel fornire una guida pratica e una comprensione pratica dell'utilizzo corretto della libreria *CAPTCHA_G*,
- Gli sviluppatori possono trarre vantaggio dalla *web-app* dimostrativa per comprendere le migliori pratiche di integrazione e configurazione della libreria *CAPTCHA_G* nelle proprie applicazioni,
- La *web-app* dimostrativa aiuta a diffondere la consapevolezza e la comprensione dell'importanza della sicurezza delle applicazioni web e dell'utilizzo adeguato della libreria *CAPTCHA_G* come misura di protezione.

1.3 Struttura del documento di analisi

Il presente documento di analisi sulle vulnerabilità della libreria *CAPTCHA_G* e della sua *web-app* dimostrativa è strutturato per fornire una panoramica chiara e dettagliata delle vulnerabilità identificate e delle relative implicazioni. La struttura del documento è organizzata come segue:

- **Sezione 1: Descrizione della libreria *CAPTCHA_G*:** Questa sezione offre una visione d'insieme della libreria *CAPTCHA_G*, compresi il suo scopo, il funzionamento e le tecnologie utilizzate. Sarà esaminato anche l'uso previsto della libreria all'interno delle applicazioni.
- **Sezione 2: Metodologia di analisi:** Qui viene presentata l'approccio seguito per identificare le vulnerabilità, inclusi gli strumenti utilizzati e l'ambiente di test impiegato durante l'analisi.



- **Sezione 3: Analisi delle vulnerabilità:** Questa sezione si concentra sull'identificazione e la descrizione delle vulnerabilità individuate nella libreria *CAPTCHA_G* e nella sua *web-app* dimostrativa. Ogni vulnerabilità sarà valutata in termini di impatto potenziale e rischio associato.
- **Sezione 4: Conclusioni:** In questa sezione vengono riassunti i risultati dell'analisi, evidenziando l'importanza delle vulnerabilità identificate e fornendo ulteriori considerazioni e suggerimenti.
- **Sezione 5: Raccomandazioni:** Qui vengono fornite raccomandazioni per mitigare le vulnerabilità identificate e per una corretta configurazione della libreria *CAPTCHA_G*. Saranno inclusi suggerimenti pratici e *best practices* per migliorare la sicurezza complessiva.
- **Sezione 6: Riferimenti:** In questa sezione sono elencate le fonti utilizzate per l'analisi inclusa la documentazione della libreria *CAPTCHA_G* e altri riferimenti rilevanti.

2 Descrizione della libreria *clock-captcha*

2.1 Scopo e funzionamento della libreria *clock-captcha*

La libreria *clock-captcha* è un prodotto open source progettato per la generazione e la verifica di $CAPTCHA_G$, nonché per fornire un servizio completo per la visualizzazione dei $CAPTCHA_G$ e la raccolta dei tentativi di soluzione. Il suo obiettivo principale è contrastare gli accessi ai servizi da parte di programmi automatizzati, noti come *bot*, riducendo gli attacchi di tipo DOS, *spam* e limitando l'accesso di *crawler* e *spider*.

La libreria *clock-captcha* consente l'inserimento di un modulo HTML all'interno delle pagine *web*, il quale visualizza un $CAPTCHA_G$ generato nel *backend*. La responsabilità di gestire questo modulo HTML ricade sulla classe `ClockCAPTCHAView`, che necessita di effettuare una richiesta al *backend* per ricevere la sfida. A sua volta, la classe `ClockCAPTCHA` nel *backend* si occupa di verificare se la richiesta è autorizzata, generando un'ora da indovinare e un *token* associato al $CAPTCHA_G$.

La libreria utilizza la crittografia AES per rendere illeggibili all'utente i dati trasmessi durante la verifica del $CAPTCHA_G$. L'immagine del $CAPTCHA_G$ vero e proprio è generata dalla classe `ClockCAPTCHAGenerator`, che riceve un oggetto `ClockImageGeneratorStrategy` per produrre un'immagine raffigurante un orologio analogico. Opzionalmente, possono essere applicate fino a due tipologie di disturbo utilizzando i *design patterns* `NoiseDecorator` e `ShapeDecorator`. Questi permettono di regolare il grado e la presenza del disturbo nell'immagine del $CAPTCHA_G$ generata.

Attraverso l'utilizzo della libreria *clock-captcha*, i servizi *web* possono implementare un sistema di $CAPTCHA_G$ efficace per prevenire l'accesso non autorizzato da parte di *bot* e altre attività malevole. Proteggendo contro gli attacchi di tipo DOS, *spam* e limitando l'accesso di *crawler* e *spider*, la libreria contribuisce a migliorare la sicurezza e l'integrità dei servizi *online*.

2.2 Tecnologie utilizzate

Sinteticamente elencate le tecnologie utilizzate per lo sviluppo della libreria:

- Javascript
- HTML
- CSS
- Typescript
- npm
- Bcrypt
- node-canvas

Per informazioni più dettagliate si rimanda al documento di Specifica Tecnica

3 Metodologia di analisi

3.1 Approccio seguito per identificare le vulnerabilità

L'analisi delle vulnerabilità della libreria *CAPTCHA_G* richiede un approccio sistematico che coinvolge diversi aspetti. Di seguito sono descritti i passi seguiti per condurre tale analisi:

1. **Esplorazione:** Familiarizzati con il funzionamento generale della libreria *CAPTCHA_G*. Studia la documentazione ufficiale, leggi gli articoli di ricerca e cerca di comprendere le tecniche di base utilizzate dalla libreria.
2. **Analisi delle minacce:** Identifica le minacce potenziali che potrebbero essere sfruttate contro la libreria *CAPTCHA_G*. Ciò può includere attacchi di forza bruta, attacchi di *reverse-engineering*, attacchi di intelligenza artificiale, attacchi basati sulle caratteristiche specifiche del *CAPTCHA_G*, ecc.
3. **Esplorazione delle vulnerabilità note:** Ricercare in letteratura e nelle fonti affidabili le vulnerabilità già conosciute relative alla libreria *CAPTCHA_G* che stai analizzando. Questo può includere pubblicazioni accademiche, *report* di sicurezza, *blog* tecnici e forum specializzati.
4. **Testing manuale:** Interagisci manualmente con la libreria *CAPTCHA_G* per individuare eventuali falle o comportamenti anomali. Prova ad eseguire diversi tipi di attacchi e verifica se la libreria è in grado di rilevarli e prevenirli in modo efficace.
5. **Testing automatizzato:** Utilizza strumenti automatizzati per testare la robustezza della libreria *CAPTCHA_G*. Esistono strumenti specializzati per il *testing* dei *CAPTCHA_G* che possono eseguire test di forza bruta, analisi delle immagini, riconoscimento di caratteri, ecc.
6. **Code review:** Esamina attentamente il codice sorgente della libreria *CAPTCHA_G*. Cerca vulnerabilità comuni come input non valido, potenziali problemi di *buffer overflow* o di gestione degli errori, e valuta la correttezza delle implementazioni crittografiche utilizzate.
7. **Analisi dei dati:** Raccogli dati sulle performance e l'efficacia della libreria *CAPTCHA_G* nell'ambito delle sue funzionalità di sicurezza. Analizza i dati raccolti per identificare eventuali punti deboli o anomalie nel sistema.

È importante sottolineare che l'analisi delle vulnerabilità è un processo continuo in quanto nuove minacce e vulnerabilità possono emergere nel tempo. Pertanto, è fondamentale mantenere la libreria *CAPTCHA_G* aggiornata e adottare misure proattive per mitigare potenziali rischi.

3.2 Strumenti utilizzati durante l'analisi

Per seguire il *workflow* sopra descritto, sono stati utilizzati degli strumenti di supporto:

- **Firefox/Chrome Dev Tools**, per analizzare come viene caricato il contenuto ed effettuare *testing* manuale,
- **Visual Studio Code**, per effettuare *code review*,
- **OWASP ZAP**, per effettuare *testing* automatizzato sulla *web-app*,



- **grype**, per effettuare *testing* automatizzato su tutto il codice, cercando vulnerabilità comuni, già note e presenti nei database.
- **npm**, per controllare che le dipendenze utilizzate non soffrano di vulnerabilità note. Parzialmente ridondante rispetto al lavoro effettuato da *grype*, utilizzato ugualmente vista la velocità ed intuitività.

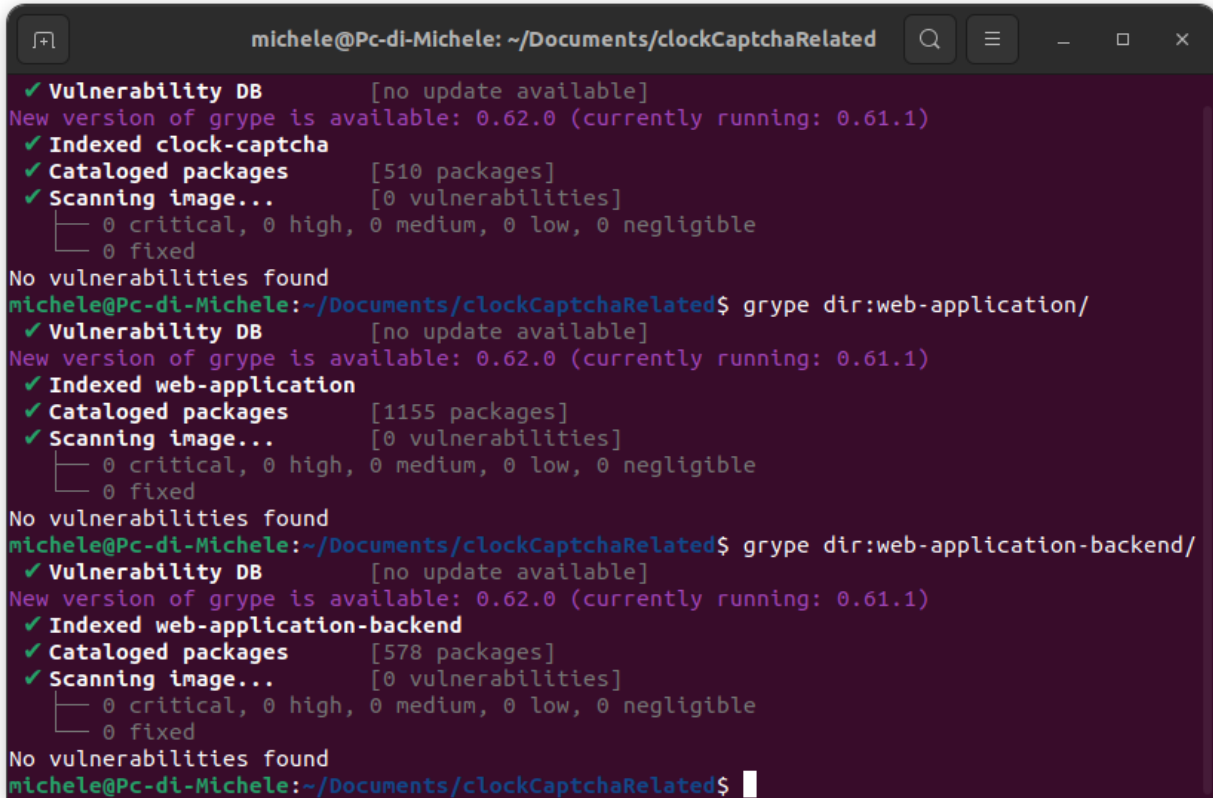
4 Analisi delle vulnerabilità

4.1 Identificazione delle possibili vulnerabilità

Nella fase di analisi delle vulnerabilità relative al servizio *clock-captcha* è fondamentale identificare tutte le possibili debolezze che potrebbero compromettere la sicurezza e l'integrità del sistema. Di seguito sono riportate le principali aree di attenzione per individuare tali vulnerabilità.

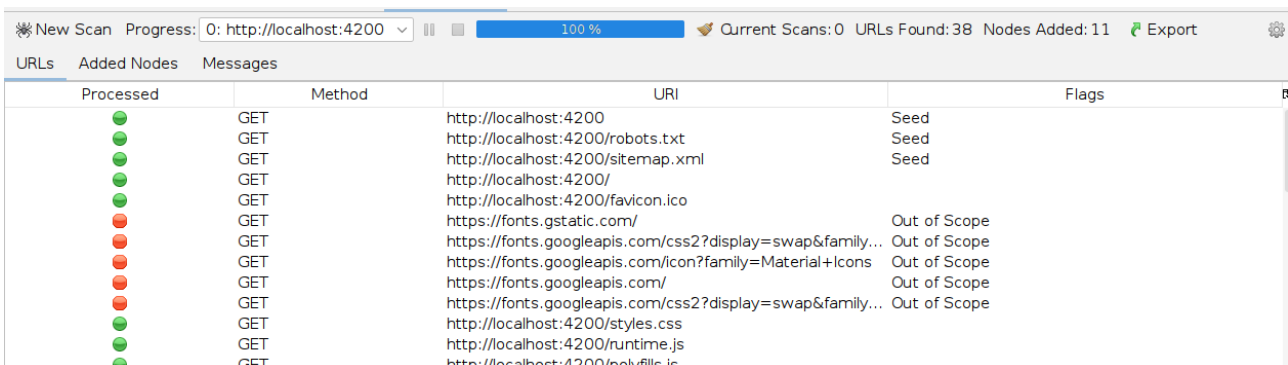
1. **Input non valido o non affidabile:** Verificare se il servizio *clock-captcha* gestisce correttamente l'input proveniente dagli utenti. Identificare eventuali mancate validazioni o controlli di sicurezza che potrebbero consentire l'inserimento di dati dannosi o potenzialmente pericolosi. Assicurarsi che l'input venga correttamente validato e filtrato per prevenire attacchi di iniezione di codice, *cross-site scripting* (XSS) o altre forme di manipolazione malevola dei dati.
2. **Errori di logica o di implementazione:** Esaminare attentamente il codice del servizio *clock-captcha* per individuare possibili errori di logica o implementazione che potrebbero essere sfruttati da attaccanti. Verificare che tutte le condizioni e i controlli siano corretti e che il flusso di esecuzione del codice sia gestito in modo sicuro. Prestare particolare attenzione a eventuali situazioni in cui le verifiche di sicurezza potrebbero essere aggirate o eluse.
3. **Vulnerabilità delle dipendenze:** Esaminare le dipendenze utilizzate dal servizio *clock-captcha* e assicurarsi che siano aggiornate e prive di vulnerabilità note. Utilizzare strumenti come *npm audit* per individuare eventuali dipendenze vulnerabili e applicare le correzioni necessarie, aggiornando le versioni o sostituendo le dipendenze con alternative più sicure.
4. **Autenticazione e autorizzazione:** Valutare i meccanismi di autenticazione e autorizzazione utilizzati dal servizio *clock-captcha*. Verificare se le identità degli utenti vengono correttamente verificate e se sono implementati i controlli di accesso appropriati. Assicurarsi che solo gli utenti autorizzati possano interagire con il servizio e che le azioni consentite siano limitate alle necessità specifiche.
5. **Esposizione di dati sensibili:** Analizzare attentamente come il servizio *clock-captcha* gestisce i dati sensibili, come le chiavi di crittografia o le informazioni di autenticazione degli utenti. Accertarsi che tali dati siano adeguatamente protetti attraverso l'uso di algoritmi di crittografia robusti, la corretta gestione delle chiavi e l'implementazione di misure di protezione per garantire l'integrità e la riservatezza dei dati.
6. **Attacchi di forza bruta o di amplificazione:** Valutare la resistenza del servizio *clock-captcha* agli attacchi di forza bruta o di amplificazione. Verificare se sono presenti meccanismi di limitazione dei tentativi di accesso e se il servizio è in grado di gestire carichi di lavoro elevati senza subire rallentamenti o degradi delle prestazioni.
7. **Esposizione di funzionalità non necessarie:** Esaminare le funzionalità del servizio *clock-captcha* e assicurarsi che solo quelle strettamente necessarie siano esposte. Limitare l'esposizione di funzionalità non utilizzate o non necessarie, in modo da ridurre potenziali punti di attacco.

Durante l'identificazione delle vulnerabilità, è importante utilizzare una combinazione di analisi statica del codice, *test* di penetrazione e revisione delle *best practices* di sicurezza per garantire un'adeguata copertura e individuare tutte le possibili debolezze. Una volta individuate le vulnerabilità, sarà possibile passare alla fase successiva di valutazione e mitigazione dei rischi.



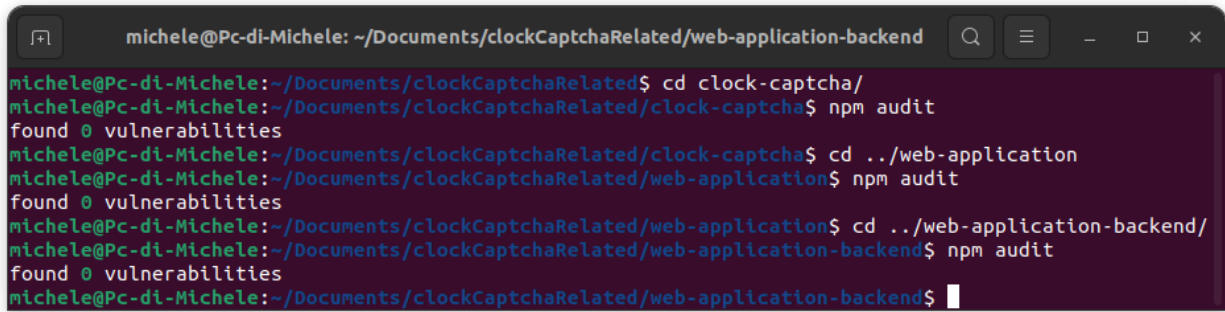
```
michele@Pc-di-Michele: ~/Documents/clockCaptchaRelated
✓ Vulnerability DB [no update available]
New version of gype is available: 0.62.0 (currently running: 0.61.1)
✓ Indexed clock-captcha
✓ Cataloged packages [510 packages]
✓ Scanning image... [0 vulnerabilities]
└─ 0 critical, 0 high, 0 medium, 0 low, 0 negligible
└─ 0 fixed
No vulnerabilities found
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated$ gype dir:web-application/
✓ Vulnerability DB [no update available]
New version of gype is available: 0.62.0 (currently running: 0.61.1)
✓ Indexed web-application
✓ Cataloged packages [1155 packages]
✓ Scanning image... [0 vulnerabilities]
└─ 0 critical, 0 high, 0 medium, 0 low, 0 negligible
└─ 0 fixed
No vulnerabilities found
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated$ gype dir:web-application-backend/
✓ Vulnerability DB [no update available]
New version of gype is available: 0.62.0 (currently running: 0.61.1)
✓ Indexed web-application-backend
✓ Cataloged packages [578 packages]
✓ Scanning image... [0 vulnerabilities]
└─ 0 critical, 0 high, 0 medium, 0 low, 0 negligible
└─ 0 fixed
No vulnerabilities found
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated$
```

Figura 1: Analisi del sorgente con gype



New Scan Progress: 0: http://localhost:4200 100 % Current Scans:0 URLs Found:38 Nodes Added: 11 Export				
URLs	Added Nodes	Messages		
Processed	Method	URI	Seed	Flags
●	GET	http://localhost:4200	Seed	
●	GET	http://localhost:4200/robots.txt	Seed	
●	GET	http://localhost:4200/sitemap.xml	Seed	
●	GET	http://localhost:4200/		
●	GET	http://localhost:4200/favicon.ico		
●	GET	https://fonts.gstatic.com/	Out of Scope	
●	GET	https://fonts.googleapis.com/css2?display=swap&family=...	Out of Scope	
●	GET	https://fonts.googleapis.com/icon?family=Material+Icons	Out of Scope	
●	GET	https://fonts.googleapis.com/	Out of Scope	
●	GET	https://fonts.googleapis.com/css2?display=swap&family=...	Out of Scope	
●	GET	http://localhost:4200/styles.css		
●	GET	http://localhost:4200/runtime.js		
●	GET	http://localhost:4200/nolvfills.js		

Figura 2: Crawling della web-app per individuare vulnerabilità - OWASP ZAP



```
michele@Pc-di-Michele: ~/Documents/clockCaptchaRelated/web-application-backend
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated$ cd clock-captcha/
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/clock-captcha$ npm audit
found 0 vulnerabilities
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/clock-captcha$ cd ../web-application
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/web-application$ npm audit
found 0 vulnerabilities
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/web-application$ cd ../web-application-backend/
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/web-application-backend$ npm audit
found 0 vulnerabilities
michele@Pc-di-Michele:~/Documents/clockCaptchaRelated/web-application-backend$
```

Figura 3: Analisi del sorgente con npm

4.2 Descrizione delle vulnerabilità individuate

Di seguito viene riportata una tabella che illustra e descrive tutte le vulnerabilità individuate durante l'analisi svolta sul prodotto descritto da questo documento:

Tabella 2: Analisi delle Vulnerabilità

Vulnerabilità	Descrizione	Impatto	Rischio
Superamento fraudolento della sfida <i>CAPTCHA_G</i> attraverso servizi di terze parti	L'attaccante può impiegare servizi di terze parti per richiedere il superamento del <i>CAPTCHA_G</i>	Medio: Consente l'accesso a strumenti automatizzati alle zone sensibili dell'applicazione o a operazioni di inserimento dati	Medio: Il <i>CAPTCHA_G</i> sviluppato non rientra negli <i>standard</i> dell'industria, rendendo difficile l'operazione.
Superamento fraudolento della sfida <i>CAPTCHA_G</i> mediante l'impiego di strumenti automatizzati.	Consiste nello sviluppo di uno strumento apposito al superamento della sfida <i>CAPTCHA_G</i> ad orologio	Medio: Consente l'accesso a strumenti automatizzati alle zone sensibili dell'applicazione o a operazioni di inserimento dati	Medio: Il <i>CAPTCHA_G</i> sviluppato non rientra negli <i>standard</i> dell'industria, rendendo difficile l'operazione.
Vulnerabilità di <i>Brute-Force Attack</i>	Consente agli attaccanti di tentare di indovinare le credenziali di autenticazione provando diverse combinazioni fino a trovare quella corretta.	Basso: possibilità di indovinare le credenziali utente o chiave segreta per utilizzare il servizio <i>clock-captcha</i>	Basso: richiede tempo e risorse significative per indovinare le credenziali. I requisiti minimi per le credenziali utente garantiscono una sicurezza adeguata. La lunghezza della chiave segreta garantisce tempi sufficientemente lunghi per indovinarla.
Vulnerabilità di <i>SQL Injection</i>	Permette agli attaccanti di inserire codice SQL malevolo nelle <i>query</i> , consentendo l'estrazione o la modifica non autorizzata dei dati nel <i>database</i> .	Elevato: compromissione dei dati sensibili e possibile divulgazione di informazioni riservate.	Basso: la vulnerabilità è correttamente gestita attraverso le <i>best-practices</i> consigliate dall'industria per le tecnologie utilizzate.

Continua nella pagina successiva

Tabella 2 – Continua dalla pagina precedente

Vulnerabilità	Descrizione	Impatto	Rischio
Vulnerabilità di <i>Cross-Site Scripting</i> (XSS)	Consente agli attaccanti di inserire <i>script</i> malevoli all'interno delle pagine <i>web</i> visualizzate dagli utenti, portando all'esecuzione di codice non autorizzato.	Medio: possibilità di furto di credenziali utente o di dirottamento delle sessioni.	Basso: Sono utilizzati provvedimenti per limitare l'impatto di tale attacco. Come corretta gestione degli <i>headers</i> e sanificazione <i>input</i> e <i>output</i> .
Vulnerabilità di <i>Remote Code Execution</i>	Consente agli attaccanti di eseguire codice arbitrario su un sistema remoto, ottenendo un controllo completo sulla macchina vulnerabile.	Elevato: compromissione totale del sistema, con potenziale per l'esecuzione di comandi dannosi.	Basso: Sono utilizzati provvedimenti per limitare l'impatto di tale attacco. Come corretta gestione degli <i>headers</i> e sanificazione <i>input</i> e <i>output</i> .
Vulnerabilità di <i>Denial of Service</i> (DoS)	Permette agli attaccanti di saturare o sovraccaricare un sistema o un'applicazione, impedendo agli utenti legittimi di accedervi o utilizzarli correttamente.	Medio: interruzione temporanea del servizio o delle funzionalità.	Medio: l'attacco richiede risorse significative ma potrebbe causare disagi e interruzioni temporanee.
Vulnerabilità di <i>Clickjacking</i>	Consente agli attaccanti di nascondere elementi o azioni dannose dietro elementi visivi benigni, portando gli utenti a fare clic su cose non desiderate o a compiere azioni non intenzionali.	Basso: azioni non autorizzate o non desiderate compiute dagli utenti.	Basso: richiede l'ingegneria sociale per ingannare gli utenti, ma l'impatto è limitato.

Continua nella pagina successiva

Tabella 2 – Continua dalla pagina precedente

Vulnerabilità	Descrizione	Impatto	Rischio
Vulnerabilità di <i>Server Side Request Forgery</i> (SSRF)	Consente agli attaccanti di indurre il <i>server</i> a effettuare richieste a risorse esterne non autorizzate, sfruttando il <i>server</i> come intermediario per attacchi a terze parti.	Medio: possibilità di accesso non autorizzato a risorse esterne o esecuzione di attacchi a terze parti.	Basso: richiede la conoscenza delle vulnerabilità specifiche e l'opportunità di indurre il <i>server</i> a effettuare richieste non autorizzate. La responsabilità ricade sullo sviluppatore di <i>web-app</i> , che deve assicurarsi di sanificare tutti gli <i>input</i> e gli <i>output</i> .
Vulnerabilità di <i>Insecure Cryptographic Storage</i>	Consente agli attaccanti di accedere a informazioni sensibili (come <i>password</i> o dati personali) memorizzate in modo non sicuro, compromettendo la loro riservatezza e integrità.	Medio: accesso non autorizzato a informazioni sensibili.	Basso: Sono utilizzate tecnologie crittografiche adatte per la trasmissione e persistenza dei dati. Se le chiavi sono custodite in modo corretto il rischio non esiste.
Vulnerabilità di <i>Remote File Inclusion</i> (RFI)	Consente agli attaccanti di includere <i>file</i> remoti in un'applicazione <i>web</i> , eseguendo codice non autorizzato o compromettendo il sistema.	Elevato: esecuzione di codice non autorizzato o compromissione del sistema.	Basso: Gli <i>input</i> sono adeguatamente sanificati per impedire questo tipo di vulnerabilità.

5 Conclusioni

L'analisi effettuata ha riscontrato alcune vulnerabilità all'interno dell'applicativo presentato. Le vulnerabilità sono però adeguatamente gestite e non rappresentano rischi elevati per la sicurezza degli utenti o dei fornitori di servizi.

Le vulnerabilità prominenti riguardano la sicurezza della sfida *CAPTCHA_G*. Purtroppo, in questo momento storico, l'intelligenza artificiale è in grado di produrre risultati eccezionali in ambito di riconoscimento di concetti complessi. È perciò prevedibile che chiunque vi dedichi sufficiente tempo e risorse, potrà sviluppare un sistema in grado di riconoscere l'ora. Inoltre le sfide *CAPTCHA_G* sono aggirabili quando, un servizio automatizzato si serve di esseri umani per la risoluzione del test, come sappiamo essere possibile attraverso servizi di terze parti come ad esempio [2captcha](#).

Nonostante ciò, *clock-captcha* resta una libreria utile per sperimentare un diverso sistema *CAPTCHA_G*. Inoltre, Essendo una libreria *open-source* eventuali vulnerabilità sfuggite al *team* di sviluppo potrebbero essere scovate dalla *community* che fruisce del servizio.

6 Raccomandazioni

6.1 Suggerimenti per mitigare le vulnerabilità identificate

- **Custodire con cautela la chiave segreta per l'uso della libreria:** senza che ciò accada, si rischia l'utilizzo non autorizzato del servizio *clock-captcha*.
- **Custodire con cautela le credenziali per Database:** avrebbe un impatto disastroso se fosse disponibile ad un attaccante. Utilizzare una password robusta e custodirla in modo sicuro.
- **Utilizzare HTTPS:** crittografando il traffico si proteggono gli utenti e gli erogatori di servizio al meglio.
- **Configurare i parametri di disturbo correttamente:** senza un sufficiente disturbo all'interno dell'orologio *CAPTCHA_G*, risulterebbe più facile per strumenti automatizzati superare il *test*.

6.2 Indicazioni per una corretta configurazione della libreria *CAPT-CHA_G*

Per delle istruzioni più precise riguardo alla configurazione si rimanda al documento "Manuale Utente". Nonostante ciò alcune precisazioni per una configurazione sicura meritano di essere affrontate più approfonditamente.

- **Non rendere pubblico per nessun motivo il file `.env`:** contiene informazioni sensibili e va pertanto protetto. Includerlo in un eventuale file `.gitignore`
- **Cercare il corretto livello di disturbo:** nonostante l'idea di disturbare in modo pesante il test *CAPTCHA_G* sembri allettante, questo finirebbe per tagliare fuori anche gli utenti umani.

7 Riferimenti

7.1 Fonti utilizzate per l'analisi

- [grype](#)
- [OWASP ZAP](#)
- [XSS prevention cheat sheet - OWASP](#)
- [RFI - Imperva](#)
- [SQL Injection prevention - OWASP](#)
- [Nodejs Security - OWASP](#)
- [Input Validation - OWASP](#)
- [Injection Prevention - OWASP](#)
- [Cryptographic storage - OWASP](#)
- [Vulnerable Dependency management - OWASP](#)
- [User Privacy Protection - OWASP](#)

7.2 Documentazione della libreria CAPTCHA e dell'applicazione

- [Repo clock-captcha](#)
- [Repo web-application](#)
- [Repo web-application-backend](#)