Web application

Checking SSL/TLS with `testssl`

`testssl.sh` is a free, open-source command-line tool designed for testing SSL/TLS vulnerabilities and misconfigurations on servers. It operates on various platforms without requiring special dependencies beyond a shell and OpenSSL, making it highly portable and easy to use for security auditing of web servers and other network services that rely on SSL/TLS for encryption and secure communication. It can assess a server's support for ciphers, protocols, certificate details, and various security-related features, providing a comprehensive overview of the server's SSL/TLS security posture.

Running `testssl` on the the web server yielded a good result:

- only TLS versions >= 1.2 are offered
- only strong cipher suites are offered with Perfect Forward Secrecy and no CBC ciphers

For detailed information, see the following output of `testssl`:

```
 Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   not offered
 ALPN/HTTP2 h2, http/1.1 (offered)

 Testing cipher categories

 NULL ciphers (no encryption)                not offered (OK)
 Anonymous NULL Ciphers (no authentication)  not offered (OK)
 Export ciphers (w/o ADH+NULL)               not offered (OK)
 LOW: 64 Bit + DES, RC[2,4] (w/o export)     not offered (OK)
 Triple DES Ciphers / IDEA                   not offered
 Obsolete CBC ciphers (AES, ARIA etc.)       not offered
 Strong encryption (AEAD ciphers)            offered (OK)


 Testing robust (perfect) forward secrecy, (P)FS -- omitting Null Authentication/Encryption, 3DES, RC4

 PFS is offered (OK)          TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 ECDHE-RSA-AES256-GCM-SHA384 DHE-RSA-AES256-GCM-SHA384
                              ECDHE-RSA-CHACHA20-POLY1305 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256
 Elliptic curves offered:     prime256v1 secp384r1 secp521r1 X25519
 DH group offered:            Unknown DH group (2048 bits)

 Testing server preferences

 Has server cipher order?     yes (OK) -- only for < TLS 1.3
 Negotiated protocol          TLSv1.3
 Negotiated cipher            TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Cipher order
    TLSv1.2:   ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-CHACHA20-POLY1305 DHE-RSA-AES256-GCM-SHA384
```

```
 Testing server defaults (Server Hello)

 TLS extensions (standard)    "status request/#5" "session ticket/#35" "renegotiation info/#65281" "EC point formats/#11" "supported versions/#43"
                              "key share/#51" "extended master secret/#23" "application layer protocol negotiation/#16"
 Session Ticket RFC 5077 hint 604800 seconds but: PFS requires session ticket keys to be rotated < daily !
 SSL Session ID support       yes
 Session Resumption           Tickets: yes, ID: yes
 TLS clock skew               Random values, no fingerprinting possible
 Signature Algorithm          SHA256 with RSA
 Server key size              RSA 2048 bits
 Server key usage             Digital Signature, Key Encipherment
 Server extended key usage    TLS Web Server Authentication, TLS Web Client Authentication
 Serial                       04714AD5135A001813360ED557C25A82D18F (OK: length 18)
 Fingerprints                 SHA1 8A39936111DD486CD4F83D25ADEE1921AA36E84A
                              SHA256 40433743F322D64CE528C95AFC110D7059DFA139C25FD3CCBBE6ADFC37402627
 Common Name (CN)             idp.dashboard.eversion.tech  (CN in response to request w/o SNI: no-sni.vercel-infra.com )
 subjectAltName (SAN)         idp.dashboard.eversion.tech
 Issuer                       R3 (Let's Encrypt from US)
 Trust (hostname)             Ok via SAN (SNI mandatory)
 Chain of trust               Ok
 EV cert (experimental)       no
 ETS/"eTLS", visibility info  not present
 Certificate Validity (UTC)   62 >= 30 days (2024-02-26 23:46 --> 2024-05-26 23:46)
 # of certificates provided   2
 Certificate Revocation List  --
 OCSP URI                     http://r3.o.lencr.org
 OCSP stapling                offered, not revoked
 OCSP must staple extension   --
 DNS CAA RR (experimental)    not offered
 Certificate Transparency     yes (certificate extension)


 Testing HTTP header response @ "/"

 HTTP Status Code             200 OK
 HTTP clock skew              +1122541 sec from localtime
 HTTP Age, RFC 7234           1122541
 Strict Transport Security    730 days=63072000 s, just this domain
 Public Key Pinning           --
 Server banner                Vercel
 Application banner           --
 Cookie(s)                    (none issued at "/")
 Security headers             Access-Control-Allow-Origin: *
                              Cache-Control: public, max-age=0, must-revalidate
 Reverse Proxy banner         --
```

```
Testing vulnerabilities

Heartbleed (CVE-2014-0160)                not vulnerable (OK), no heartbeat extension
CCS (CVE-2014-0224)                       not vulnerable (OK)
Ticketbleed (CVE-2016-9244), experiment.  test failed around line 14455 (debug info: 48, 5454502F30)
ROBOT                                     Server does not support any cipher suites that use RSA key transport
Secure Renegotiation (RFC 5746)           supported (OK)
Secure Client-Initiated Renegotiation     not vulnerable (OK)
CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
BREACH (CVE-2013-3587)                    potentially NOT ok, "br" HTTP compression detected. - only supplied "/" tested
                                          Can be ignored for static pages or if no secrets in the page
POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below TLS 1.2 offered
SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
FREAK (CVE-2015-0204)                     not vulnerable (OK)
DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                          make sure you don't use this certificate elsewhere with SSLv2 enabled services
                                          https://search.censys.io/search?resource=hosts&virtual_hosts=INCLUDE&q=40433743F322D64CE528C95AFC110D7059DFA139C25FD3CCBBE6ADFC37402627
LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK): no DH EXPORT ciphers, no common prime detected
BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
LUCKY13 (CVE-2013-0169), experimental     not vulnerable (OK)
RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)


Testing 370 ciphers via OpenSSL plus sockets against the server, ordered by encryption strength

Hexcode  Cipher Suite Name (OpenSSL)       KeyExch.   Encryption  Bits     Cipher Suite Name (IANA/RFC)
-----------------------------------------------------------------------------------------------------------------------
 x1302   TLS_AES_256_GCM_SHA384            ECDH 253   AESGCM      256      TLS_AES_256_GCM_SHA384
 x1303   TLS_CHACHA20_POLY1305_SHA256      ECDH 253   ChaCha20    256      TLS_CHACHA20_POLY1305_SHA256
 xc030   ECDHE-RSA-AES256-GCM-SHA384       ECDH 253   AESGCM      256      TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
 x9f     DHE-RSA-AES256-GCM-SHA384         DH 2048    AESGCM      256      TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
 xcca8   ECDHE-RSA-CHACHA20-POLY1305       ECDH 253   ChaCha20    256      TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
 x1301   TLS_AES_128_GCM_SHA256            ECDH 253   AESGCM      128      TLS_AES_128_GCM_SHA256
 xc02f   ECDHE-RSA-AES128-GCM-SHA256       ECDH 253   AESGCM      128      TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

```
Running client simulations (HTTP) via sockets

 Android 6.0                TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 7.0 (native)       TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 8.1 (native)       TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 253 bit ECDH (X25519)
 Android 9.0 (native)       TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Android 10.0 (native)      TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Android 11 (native)        TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Android 12 (native)        TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Chrome 79 (Win 10)         TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Chrome 101 (Win 10)        TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Firefox 66 (Win 8.1/10)    TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Firefox 100 (Win 10)       TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 IE 6 XP                    No connection
 IE 8 Win 7                 No connection
 IE 8 XP                    No connection
 IE 11 Win 7                TLSv1.2 DHE-RSA-AES256-GCM-SHA384, 2048 bit DH
 IE 11 Win 8.1              TLSv1.2 DHE-RSA-AES256-GCM-SHA384, 2048 bit DH
 IE 11 Win Phone 8.1        No connection
 IE 11 Win 10               TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Edge 15 Win 10             TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 253 bit ECDH (X25519)
 Edge 101 Win 10 21H2       TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Safari 12.1 (iOS 12.2)     TLSv1.3 TLS_CHACHA20_POLY1305_SHA256, 253 bit ECDH (X25519)
 Safari 13.0 (macOS 10.14.6) TLSv1.3 TLS_CHACHA20_POLY1305_SHA256, 253 bit ECDH (X25519)
 Safari 15.4 (macOS 12.3.1) TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Java 7u25                  No connection
 Java 8u161                 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Java 11.0.2 (OpenJDK)      TLSv1.3 TLS_AES_128_GCM_SHA256, 256 bit ECDH (P-256)
 Java 17.0.3 (OpenJDK)      TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 go 1.17.8                  TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 LibreSSL 2.8.3 (Apple)     TLSv1.2 ECDHE-RSA-CHACHA20-POLY1305, 253 bit ECDH (X25519)
 OpenSSL 1.0.2e             TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 OpenSSL 1.1.0l (Debian)    TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 253 bit ECDH (X25519)
 OpenSSL 1.1.1d (Debian)    TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 OpenSSL 3.0.3 (git)        TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)
 Apple Mail (16.0)          TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Thunderbird (91.9)         TLSv1.3 TLS_AES_128_GCM_SHA256, 253 bit ECDH (X25519)

Done 2024-03-25 12:48:42 [0102s] -->> 76.76.21.21:443 (idp.dashboard.eversion.tech) <<--
```

Scanning for hidden directories/subdomains with `gobuster`

Gobuster is a powerful open-source tool designed for reconnaissance and enumeration during penetration testing and security assessments. It efficiently scans web applications and directories, aiding in the discovery of hidden files, directories, and vulnerabilities. Gobuster utilizes brute-force techniques, such as dictionary-based attacks, to uncover sensitive information that might be accessible to attackers. With its versatility and speed, Gobuster is an essential asset for security professionals aiming to identify potential entry points and strengthen defenses against cyber threats.

Scanning for hidden directories

No hidden directory found.

Scanning for hidden subdomains

```
to.nguyentruongan@2BY1-1GC6N ~ % gobuster dns -d eversion.tech -w SecLists/Discovery/DNS/subdomains-top1million-110000.txt --wildcard

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Domain:             eversion.tech
[+] Threads:            10
[+] Wildcard forced:    true
[+] Timeout:            1s
[+] Wordlist:           SecLists/Discovery/DNS/subdomains-top1million-110000.txt
===============================================================
Starting gobuster in DNS enumeration mode
===============================================================
Found: autodiscover.eversion.tech

Found: test.eversion.tech

Found: api.eversion.tech

Found: www.test.eversion.tech

Found: api2.eversion.tech

Progress: 114441 / 114442 (100.00%)
===============================================================
Finished
===============================================================
```

Countermeasures:

- Verify whether the hidden directories or subdomains are still active and maintained. If not, remove or update them to maintain a clean and secure web presence.

Content Security Policy

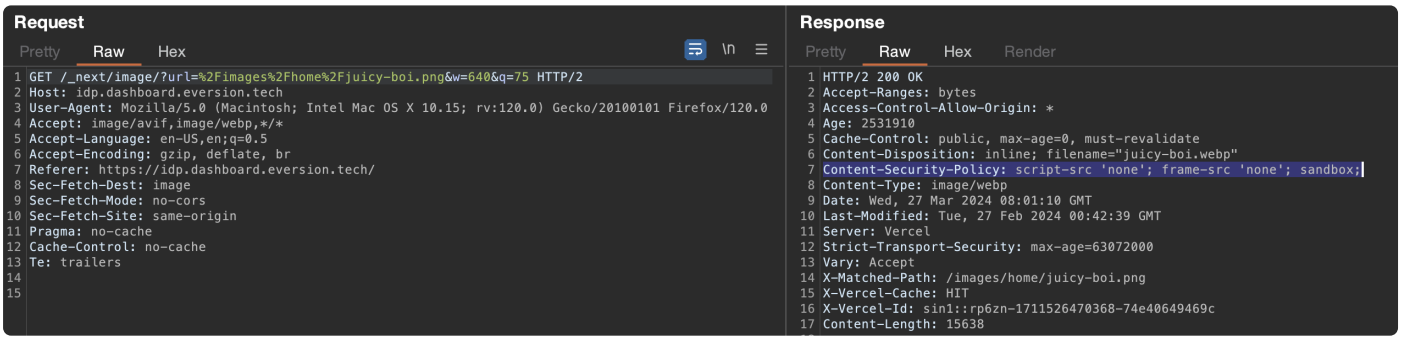
Content Security Policy (CSP) is a security standard introduced to help prevent cross-site scripting (XSS), clickjacking, and other code injection attacks resulting from execution of malicious content in the trusted web page context. CSP is normally implemented through a web server sending the Content-Security-Policy HTTP header. CSP is composed of a series of directives that specify types of resources and their allowed sources. Some of the key directives include:

- `default-src`: Serves as a fallback for other source directives that are not explicitly set in the policy.
- `script-src`: Defines valid sources for JavaScript.
- `style-src`: Defines valid sources for stylesheets.
- `img-src`: Defines valid sources for images.
- `connect-src`: Defines valid sources for XMLHttpRequest, WebSockets, and EventSource.
- `font-src`: Defines valid sources for fonts.
- `frame-src`: Defines valid sources for frames and iframes.
- `report-uri` / `report-to`: Specifies where to send reports about policy violations.

Sources can be specified using URLs, keywords like `'self'` (to allow resources from the same origin as the document), `'none'` (to disallow resources of the given type entirely), or `'unsafe-inline'` / `'unsafe-eval'` (to allow the use of inline resources or eval, respectively, though these weaken the policy).

Finding:

The web application uses: `Content-Security-Policy: script-src 'none'; frame-src 'none'; sandbox;`

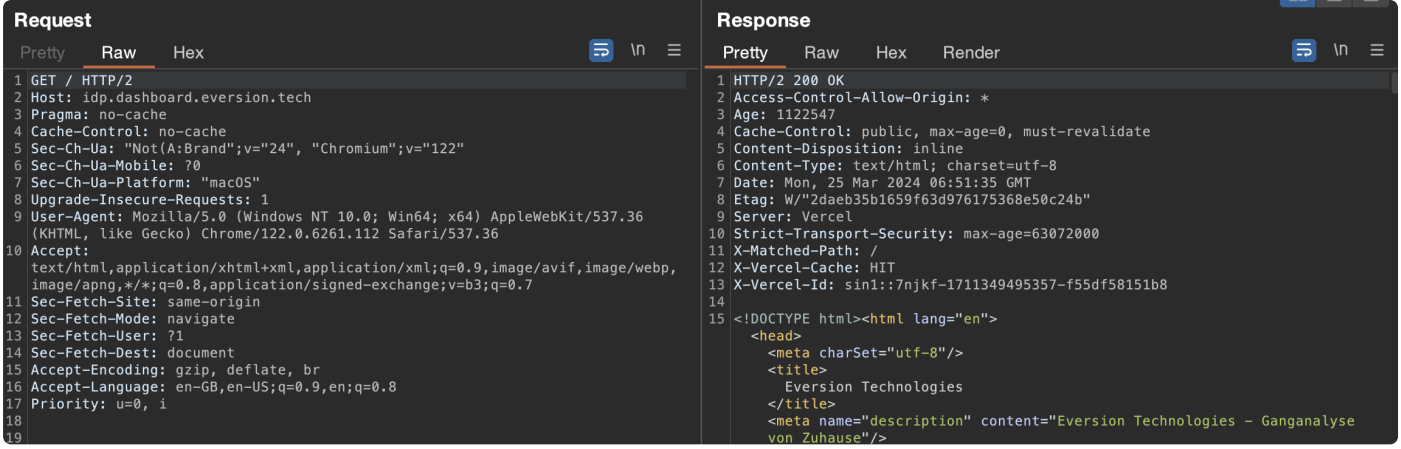


This CSP header includes:

1. **`script-src 'none';`** This directive specifies that no scripts are allowed to be executed on the page. The `'none'` value effectively blocks all script execution. This means that the page cannot execute inline scripts, nor can it load scripts from external sources.
2. **`frame-src 'none';`** This directive restricts the URLs which can be loaded using frame or iframe elements on the page. By specifying `'none'`, it disallows all content to be framed.
3. **`sandbox;`** The `sandbox` directive applies extra restrictions to the content in the browsing context. Without any parameters, it is equivalent to setting an empty value, which applies all restrictions. These restrictions include, but are not limited to, preventing forms from being submitted, blocking plugins, preventing script execution, and blocking top-level navigation to a different site. It essentially treats the loaded page as if it were opened in a unique origin, thus severely limiting what it can do.

However, it is highly recommended to also set the `object-src` directive to `'none'` in the Content Security Policy (CSP) because we might want to prevent the loading of all types of plugin content, including Flash, Java applets, Silverlight, and others that could execute JavaScript. Plugins like these have historically been a vector for security vulnerabilities, including enabling malicious actors to execute JavaScript within the security context of the site.

Moreover, the consistency of the CSP header configuration is lacking across all requests. Specifically, some responses, like the one detailed below, were absent of the CSP header. It's advisable to uniformly apply the CSP header across all server responses for enhanced security.



Countermeasures:

1. Set the `object-src` directive to `'none'` in the Content Security Policy
2. Make the CSP header consistent across all requests

Click-jacking attacks

Click-jacking is a malicious technique where an attacker tricks a user into clicking on a deceptive or invisible element on a webpage. The attacker overlays a legitimate webpage with a transparent layer, hiding malicious buttons or links underneath. When the user interacts with the visible content, they unwittingly trigger actions on the hidden elements, potentially leading to unintended consequences such as installing malware, divulging sensitive information, or performing unauthorized transactions. This technique exploits the trust users have in familiar websites and can be used for various nefarious purposes.

Finding:

The web application is susceptible to click-jacking attacks.

Proof of Concept (PoC):

clickjacked.html

Countermeasures:

1. Utilizing the "frame-ancestors" directive in Content Security Policy (CSP). For example:

- Content-Security-Policy: frame-ancestors 'none';
- Content-Security-Policy: frame-ancestors 'self';
- Content-Security-Policy: frame-ancestors normal-website.com;

2. Implementing the X-Frame-Options Header. For example:

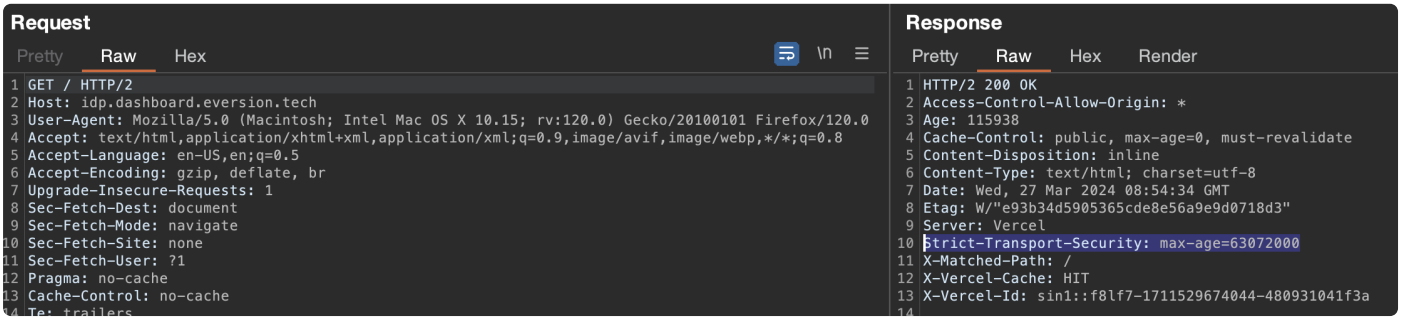
- X-Frame-Options: deny
- X-Frame-Options: sameorigin
- X-Frame-Options: allow-from https://normal-website.com

HTTP Strict-Transport-Security Header

The HTTP Strict-Transport-Security (HSTS) header is a security mechanism that instructs web browsers to interact with a website only over secure HTTPS connections, even if the user attempts to access it via HTTP. This helps prevent various types of attacks, such as man-in-the-middle attacks, by enforcing encryption and ensuring data integrity during transit.

Finding:

The web application uses: `Strict-Transport-Security: max-age=63072000` which instructs web browsers to enforce HTTPS connections exclusively for the specified duration of 63072000 seconds (approximately 2 years). However, it is highly recommend to also use the use the `includeSubDomains` directive to extend the HSTS policy to all subdomains of the website, in case there will be any new subdomains of the website in the future.



Missing account deletion feature

Finding:

The web application's dashboard currently does not include a feature enabling users to delete their own account and associated data. It's advisable to consider incorporating this functionality, as certain security standards and regulations may necessitate it.

Countermeasures:

Consider implementing the account deletion feature.

Inclusion of external scripts

Incorporating external scripts into the system might introduce potential security vulnerabilities, as external scripts could be compromised or altered maliciously, leading to various security threats

such as cross-site scripting (XSS) attacks, data breaches, or unauthorized access to sensitive information.

Finding:

The web application uses external scripts from `code.tidio.co` without additional protection in place.

Countermeasures:

1. **Subresource Integrity (SRI):** Implement Subresource Integrity (SRI) to mitigate the risks associated with the inclusion of external scripts. SRI ensures that resources, like scripts or stylesheets, are delivered unchanged from the original source. By verifying the integrity of external scripts, SRI helps prevent malicious modifications or tampering, thereby enhancing the security posture of the system.
2. **Iframe with Sandbox Attribute:** Utilize iframes with the 'sandbox' attribute to confine the execution environment of embedded content. The 'sandbox' attribute restricts various capabilities of the iframe, such as script execution, form submission, and navigational abilities, thereby reducing the impact of potential security exploits. By isolating the embedded content within a secure sandbox environment, the system can effectively mitigate the risks associated with malicious external scripts.

Weak password policy

The weak password policy arises when an organization or system does not enforce strong password creation guidelines, leaving user accounts vulnerable to unauthorized access through common attacks such as password guessing, brute force, or dictionary attacks. This lack of stringent policies allows users to create simple, easily guessable passwords that can be quickly compromised by attackers, leading to potential data breaches, identity theft, and unauthorized access to sensitive information. Implementing a strong password policy, which includes requirements for password complexity, length, expiration, and uniqueness, is crucial in safeguarding against these vulnerabilities and enhancing overall security posture.

Finding:

The web application allows for weak passwords such as `12345678` or `password` which could be easily compromised.

Countermeasures:

Enforce strong password policy which might include:

1. **Minimum Length**: Passwords must be at least 12 characters long. This length helps protect against brute-force attacks.

2. **Complexity Requirements**: Passwords must contain at least:
    - One uppercase letter (A-Z)
    - One lowercase letter (a-z)
    - One number (0-9)
    - One special character (e.g., !, @, #, $, etc.)
3. **No Personal Information**: Passwords should not contain easily accessible personal information, such as user names, real names, company names, or dates of birth, which can be guessed or found through social engineering.
4. **No Sequential or Repetitive Characters**: Passwords must not include sequences or repeated characters (e.g., 123456, aaaa, abcdef).
5. **Expiration and Rotation**: Passwords must be changed e.g. every 90 days, and the new password cannot be the same as any of the last four passwords used. This rule helps mitigate the risk of long-term exposure if a password is somehow compromised.
6. **Account Lockout Policy**: After five consecutive incorrect attempts, the account should be locked for a period of time (e.g., 15 minutes) or until an administrator unlocks it. This policy helps prevent brute force attacks.

Email/Username enumeration with account lockout

Email/Username Enumeration with Account Lockout refers to a security vulnerability where an attacker can determine if an email address or username exists on a system due to the way login failures are handled. When a system locks an account after a certain number of failed login attempts, it may display different responses for valid and invalid usernames or emails. For instance, an attacker attempting to log in with various emails may receive a message stating that the account has been locked after several attempts for a valid email, whereas an attempt with an invalid email might simply state that the username or password is incorrect. This difference allows attackers to infer which emails or usernames are registered in the application, potentially leading to targeted attacks or unauthorized access.

Finding:

The web application reveals distinct error messages after five login attempts with a valid email, enabling an attacker to ascertain the existence of this email within the web application.

Countermeasures:

1. **Uniform Error Messages**: Ensure that the error messages displayed after failed login attempts are consistent, regardless of whether the email is valid or not. For example, use a generic message like "Incorrect login details or the account has been locked due to multiple failed attempts. Please try again later or contact support."
2. **Delay and Lockout Policies**: Implement a delay in response time after a certain number of failed attempts, followed by a lockout policy that is uniformly enforced, making it less practical

for attackers to use this method for enumeration. The policy should apply the same action regardless of the account's existence.

3. **Monitoring and Alerts**: Set up monitoring for multiple failed login attempts and alert system administrators of such activities. This can help in identifying and mitigating enumeration attacks early.
4. **CAPTCHA Integration**: Introduce CAPTCHA challenges after a series of failed login attempts to prevent automated scripts from rapidly testing email addresses, thereby protecting against automated enumeration attempts.
5. **Rate Limiting**: Implement rate limiting to control the number of login attempts allowed from a single IP address over a certain period, reducing the feasibility of enumeration attacks.
6. **Multi-Factor Authentication (MFA)**: Enforcing MFA can add an additional layer of security, making it significantly more difficult for attackers to gain unauthorized access, even if they successfully determine an email address associated with an account.

Outdated software Version

Outdated software versions refer to instances where software applications or systems are running on older, potentially unsupported versions that lack the latest updates, patches, and security fixes. These outdated versions pose significant security risks, as they may contain known vulnerabilities that could be exploited by attackers to compromise the integrity, confidentiality, or availability of the system. Upgrading to the latest software versions is essential for maintaining a secure and resilient computing environment, as it ensures that critical security patches are applied, reducing the likelihood of successful cyberattacks and data breaches.

Finding:

The web application uses the following outdated software library:

- **nextjs 12.3.4**, which has the following vulnerability: CVE-2023-46298

Countermeasures:

Update the softwares in use.

Information Disclosure

Information disclosure can be highly valuable to an attacker for several reasons:

1. **Attack Surface Analysis**: Knowing what information is publicly available about a target system or organization allows attackers to assess the potential attack surface. This helps them identify potential vulnerabilities, weak points, or avenues for exploitation.
2. **Exploiting Vulnerabilities**: Information disclosure often reveals details about the software, hardware, or configurations used by the target. This information can be crucial for identifying

specific vulnerabilities or misconfigurations that can be exploited to gain unauthorized access or execute attacks.

3. **Social Engineering**: Information disclosed about individuals within an organization, such as their roles, responsibilities, or contact details, can be exploited for social engineering attacks. Attackers can use this information to craft convincing phishing emails, impersonate trusted individuals, or manipulate targets into revealing sensitive information.
4. **Building Target Profiles**: Aggregated information from multiple sources of disclosure can help attackers build detailed profiles of target organizations or individuals. This includes information about infrastructure, technologies, personnel, business processes, and even personal habits, which can aid in crafting highly targeted and effective attacks.
5. **Reconnaissance for Future Attacks**: Information disclosure is often part of the reconnaissance phase of an attack. By gathering as much information as possible about the target, attackers can plan and execute more sophisticated and targeted attacks in the future, potentially with greater success and less chance of detection.

Overall, information disclosure provides attackers with valuable insights and resources that can be leveraged to launch various types of cyber attacks, ranging from relatively simple exploits to highly sophisticated and targeted campaigns.

Finding:

HTTP response headers include name of the server in use: `Server: Vercel`