Probely

ISO 27001 COMPLIANCE REPORT

PROD of test.anilyekta - https://test.anilyekta.com.tr:8443

Report generated on Sep. 22, 2024 at 13:11 UTC

Summary

This section contains the scan summary

TARGET https://test.anilyekta.com.tr:8443

Report generated on Sep. 22, 2024 at 13:11 UTC

STARTED	ENDED	DURATION	SCAN PROFILE
Sep. 14, 2024, 15:23 UTC	Sep. 14, 2024, 15:45 UTC	22 minutes	Normal

NUMBER OF FINDINGS

	CURRENT SCAN	FROM LAST SCAN	PENDING FIX
HIGH	1	= 0	1
MEDIUM	2	= 0	1
LOW	5	= 0	5

TOP 5

HSTS header not enforced	1
Missing clickjacking protection	1
Missing Content Security Policy header	1
Browser content sniffing allowed	1
Referrer policy not defined	1

ISO 27001 REQUIREMENTS CHECKLIST

		TESTED	PASSED
A.5.14	Information transfer	✓	✗
A.5.33	Protection of records	✓	✗
A.5.34	Privacy and protection of personally identifiable information (PII)	✓	✗
A.8.2	Privileged access rights	✓	✓
A.8.3	Information access restriction	✓	✗
A.8.4	Access to source code	✓	✓
A.8.5	Secure authentication	✓	✓
A.8.8	Management of technical vulnerabilities	✓	✓
A.8.9	Configuration management	✓	✗
A.8.12	Data leakage prevention	✓	✗
A.8.15	Logging	✓	✓
A.8.24	Use of cryptography	✓	✗
A.8.25	Secure development life cycle	✓	✗

A.8.26	Application security requirements	✓	✗
A.8.27	Secure system architecture and engineering principles	✓	✗
A.8.28	Secure coding	✓	✗
A.8.29	Security testing in development and acceptance	✓	✓

Settings

This section contains the summary of settings that were used during this scan

---

✔ LOGIN FORM

FIELD NAME	FIELD VALUE
password	****

✔ LOGOUT DETECTION

URL TO CHECK SESSION
https://test.anilyekta.com.tr/Giris/Cikis

LOGGED OUT WHEN ANY FOUND	
Url	https://test.anilyekta.com.tr

✔ SCAN PROFILE

Normal

Tests for all supported vulnerabilities. It's the recommended scanning profile since it offers the best quality/time ratio.

Technical Summary

The following table summarizes the findings, ordered by their severity

#	SEVERITY	VULNERABILITY	STATE
11	HIGH	**SQL Injection** https://test.anilyekta.com.tr/Giris/Index **password**	**NOT FIXED**
10	MEDIUM	**SSL cookie without Secure flag** https://test.anilyekta.com.tr/Giris/Index **.AspNetCore.Session**	**NOT FIXED**
6	MEDIUM	**HSTS header not enforced** https://test.anilyekta.com.tr:8443	**INVALID**
7	LOW	**Weak cipher suites enabled** https://test.anilyekta.com.tr:8443	**NOT FIXED**
3	LOW	**Referrer policy not defined** https://test.anilyekta.com.tr:8443	**NOT FIXED**
5	LOW	**Missing Content Security Policy header** https://test.anilyekta.com.tr:8443	**NOT FIXED**
4	LOW	**Missing clickjacking protection** https://test.anilyekta.com.tr:8443	**NOT FIXED**
1	LOW	**Browser content sniffing allowed** https://test.anilyekta.com.tr:8443	**NOT FIXED**

Exhaustive Test List

The following pages contain the list of vulnerabilities we tested in this scan, taking into consideration the chosen profile

---

- Reflected cross-site scripting
- Cookie without HttpOnly flag
- Open redirection
- SQL Injection
- Missing cross-site request forgery protection
- Missing clickjacking protection
- Stored cross-site scripting
- Insecure crossdomain.xml policy
- SSL cookie without Secure flag
- HTTP TRACE method enabled
- Directory Listing
- ASP.NET tracing enabled
- Path traversal
- Remote File Inclusion
- ASP.NET ViewState without MAC
- Session Token in URL
- Application error message
- Private IP addresses disclosed
- OS command injection
- XML external entity injection
- ASP.NET debugging enabled
- Insecure Silverlight clientaccesspolicy.xml policy
- PHP code injection
- Server-side JavaScript injection
- Ruby code injection
- Python code injection
- Server-side template injection
- Unencrypted communications
- HSTS header not enforced
- Mixed content
- Cross Origin Resource Sharing: Arbitrary Origin Trusted
- Certificate with insufficient key size or usage, or insecure signature algorithm
- Expired TLS certificate
- Insecure SSL protocol version 3 supported
- Deprecated TLS protocol version 1.0 supported
- Deprecated TLS protocol version 1.1 supported
- Secure TLS protocol version 1.2 not supported
- Weak cipher suites enabled
- Server Cipher Order not configured
- Untrusted TLS certificate
- Heartbleed
- Secure Renegotiation is not supported
- TLS Downgrade attack prevention not supported
- Drupal version with known vulnerabilities
- WordPress version with known vulnerabilities
- Joomla! version with known vulnerabilities

- Certificate without revocation information
- Full path disclosure
- Log file disclosure
- Backup file disclosure
- HSTS header set in HTTP
- HSTS header with low duration and no subdomain protection
- HSTS header with low duration
- HSTS header does not protect subdomains
- Inclusion of cryptocurrency mining script
- Insecure SSL protocol version 2 supported
- Browser content sniffing allowed
- Referrer policy not defined
- Insecure referrer policy
- Potential DoS on TLS Client Renegotiation
- JQuery library with known vulnerabilities
- AngularJS library with known vulnerabilities
- Bootstrap library with known vulnerabilities
- JQuery Mobile library with known vulnerabilities
- JQuery Migrate library with known vulnerabilities
- TLS certificate about to expire
- Moment.js library with known vulnerabilities
- Prototype library with known vulnerabilities
- React library with known vulnerabilities
- SWFObject library with known vulnerabilities
- TinyMCE library with known vulnerabilities
- Backbone library with known vulnerabilities
- Mustache library with known vulnerabilities
- Handlebars library with known vulnerabilities
- Dojo library with known vulnerabilities
- jPlayer library with known vulnerabilities
- CKEditor library with known vulnerabilities
- DWR library with known vulnerabilities
- Flowplayer library with known vulnerabilities
- DOMPurify library with known vulnerabilities
- Plupload library with known vulnerabilities
- easyXDM library with known vulnerabilities
- Ember library with known vulnerabilities
- YUI library with known vulnerabilities
- Sessvars library with known vulnerabilities
- prettyPhoto library with known vulnerabilities
- jQuery UI library with known vulnerabilities
- WordPress plugin with known vulnerabilities
- Invalid referrer policy
- Insecure PHP Object deserialization
- Missing Content Security Policy header
- Insecure Content Security Policy
- GraphQL Introspection enabled
- Log4Shell
- Vue.js library with known vulnerabilities
- Spring Cloud SPEL Code Injection (CVE-2022-22963)
- Spring4Shell
- Knockout library with known vulnerabilities
- Weak JWT HMAC secret
- JWT accepting none algorithm

- JWT signature is not being verified
- Using jwk parameter to verify JWTs
- JWT algorithm confusion
- MongoDB Injection
- Next.js library with known vulnerabilities
- Underscore.js library with known vulnerabilities
- Chart.js library with known vulnerabilities
- JSZip library with known vulnerabilities
- GraphQL Misconfiguration
- Insecure browser XSS protection enabled
- Hidden file found
- Svelte library with known vulnerabilities
- Axios library with known vulnerabilities
- Cookie with SameSite attribute set to None
- Server-side request forgery
- CRLF injection
- Froala library with known vulnerabilities
- Highcharts library with known vulnerabilities
- Supply Chain Compromise
- PDF.js library with known vulnerabilities
- Lodash library with known vulnerabilities
- Select2 library with known vulnerabilities
- UAParser.js library with known vulnerabilities
- MathJax library with known vulnerabilities

Detailed Finding Descriptions

This section contains the findings in more detail, ordered by severity

11 SQL Injection

HIGH

CVSS SCORE
7.7

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N

METHOD	PATH	REQUEST BODY
POST	https://test.anilyekta.com.tr/Giris/Index	password

DESCRIPTION

SQL Injections are the most common form of injections because SQL databases are very popular in dynamic web applications. This vulnerability allows an attacker to tamper existing SQL queries performed by the web application. Depending on the queries, the at tacker might be able to access, modify or even destroy data from the database.

Since databases are commonly used to store private data, such as authentication information, personal user data and site content, if an attacker gains access to it, the consequences are typically very severe, ranging from defacement of the web application to us ers data leakage or loss, or even full control of the web application or database server.

EVIDENCE

As evidence that is possible to take advantage of this vulnerability, we have extracted the following data from the database engin e:

```
DBMS: Microsoft SQL Server 2022
```

```
Databases: fastep, hsdb, moceh, telnHb
```

REQUEST

```
POST /Giris/Index HTTP/1.1
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
accept-encoding: gzip, deflate
accept-language: en-US
cache-control: no-cache
content-length: 78
content-type: application/x-www-form-urlencoded
origin: https://test.anilyekta.com.tr
pragma: no-cache
priority: u=0, i
referer: https://test.anilyekta.com.tr/
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: same-origin
sec-fetch-user: ?1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelySPDR/0.2.0
host: test.anilyekta.com.tr
username=&password=%27+OR+%271%27%3D%271%27+WAITFOR+DELAY+%270%3A0%3A10%27+--+
```

RESPONSE

```
HTTP/1.1 302 Found
Date: Sat, 14 Sep 2024 15:39:16 GMT
Connection: keep-alive
Cache-Control: no-cache,no-store
expires: -1
location: /Anasayfa
pragma: no-cache
set-cookie: .AspNetCore.Session=CfDJ8EFsZ14a709OnBUKQD3EYLn9ANGwFUaUGAHq7eoI62PE
MrIsqnSqqHZfJkCSiUkLpQYgRZZdePWih%2ByyyscKSz3hYZ5uGVJW2tej6S7COri5ABeHr6LTz4oa9M
GZLZYE%2FirUqkYC2FgKbemb5Rv6Pv%2FYZC%2FwcNVUElKhb%2FALkDoz; path=/;
samesite=lax; httponly
x-powered-by: ASP.NET
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=8v
mcApA3fAKJrfHguQWvQlKQ2N0cHMU%2BPYSItZJHDUufLZ4DgrRs%2FOuRoIYVpDm%2B14x5z2iaI%2B
mfmnYxuv79johh5%2Fwx23rPx2i%2FP6UQl0O%2B8BZZC1egl1apwZivwjiJyqnHjq3zN4o%3D"}],"g
roup":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 8c3177c23aebbf4d-DUB
alt-svc: h3=":443"; ma=86400
content-length: 0
```

10

SSL cookie without Secure flag

MEDIUM

CVSS SCORE
4.3

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

METHOD	PATH	COOKIE
POST	https://test.anilyekta.com.tr/Giris/Index	.AspNetCore.Session

DESCRIPTION

The cookie secure flag is intended to prevent browsers from submitting the cookie in any HTTP requests that use an unencrypted c onnection, thus an attacker that is eavesdropping the connection will not be able to get that cookie.

A flag without the secure flag set will always be sent on every HTTP request that matches the scope of cookie, i.e. the domain for which it is set. What this means is that if your application inadvertently makes an HTTP request (without encryption), this request will carry the cookie and any attacker that can eavesdrop the victim traffic will be able to read that cookie.

If the cookie in question is the session cookie, the attacker will be able to hijack the victim account.

EVIDENCE

The cookie being set without the Secure flag:

```
set-cookie: .AspNetCore.Session=CfDJ8EFsZ14a709OnBUKQD3EYLn5JeOl8aPyaK%2B1N78jUPD7o6wwv%2FD89ZfiVXE837vlaoCpcgvRS
xbQv7oDkYN8iNAeYzF6kMfDOCrYw0THMyQeuN3kqrqk1%2FE1k4cLJm8tEaSciD0UeQMSx%2FeJSSIkevHL5jhXUdjR4A8482y35b5q; path=/;
 samesite=lax; httponly
```

REQUEST

```
POST /Giris/Index HTTP/1.1
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
accept-encoding: gzip, deflate
accept-language: en-US
cache-control: no-cache
content-length: 23
content-type: application/x-www-form-urlencoded
origin: https://test.anilyekta.com.tr
pragma: no-cache
priority: u=0, i
referer: https://test.anilyekta.com.tr/
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: same-origin
sec-fetch-user: ?1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelySPDR/0.2.0
host: test.anilyekta.com.tr
username=&password=1997
```

RESPONSE

```
HTTP/1.1 302
alt-svc: h3=":443"; ma=86400
cache-control: no-cache,no-store
cf-cache-status: DYNAMIC
cf-ray: 8c3163a1e8e2be37-DUB
```

```
date: Sat, 14 Sep 2024 15:25:21 GMT
expires: -1
location: /Anasayfa
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
pragma: no-cache
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=HU
SwhdKNSQSNxRFcMrb40MpjiD%2FqQSMEIfj0yz1SQHHBqKbN%2FpHgwLoLxo1CcQI%2B%2BXbzbLYyLs
YDUyn3Sgc2LHTxT12VBchgj9OpN1rr1pP%2Bne4bKaXN3g9URtBqHAwat2HVbZZ7GDA%3D"}],"group
":"cf-nel","max_age":604800}
server: cloudflare
set-cookie: .AspNetCore.Session=CfDJ8EFsZ14a709OnBUKQD3EYLn5JeOl8aPyaK%2B1N78jUP
D7o6wwv%2FD89ZfiVXE837vlaoCpcgvRSxbQv7oDkYN8iNAeYzF6kMfDOCrYw0THMyQeuN3kqrqk1%2F
E1k4cLJm8tEaSciD0UeQMSx%2FeJSSIkevHL5jhXUdjR4A8482y35b5q; path=/; samesite=lax;
httponly
x-powered-by: ASP.NET
content-length: 0
```

6

HSTS header not enforced

MEDIUM

CVSS SCORE
4.2

CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N

METHOD	PATH
GET	https://test.anilyekta.com.tr:8443

DESCRIPTION

The application does not force users to connect over an encrypted channel, i.e. over HTTPS. If the user types the site address in th e browser without starting with *https*, it will connect to it over an insecure channel, even if there is a redirect to HTTPS later. Even if the user types *https*, there may be links to the site in HTTP, forcing the user to navigate insecurely. An attacker that is able to in tercept traffic between the victim and the site or spoof the site's address can prevent the user from ever connecting to it over an encrypted channel. This way, the attacker is able to eavesdrop all communications between the victim and the server, including th e victim's credentials, session cookie and other sensitive information.

EVIDENCE

Response headers, missing the `Strict-Transport-Security` header:

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7
R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc
90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
```

REQUEST

```
GET / HTTP/2
host: test.anilyekta.com.tr:8443
accept: */*
accept-encoding: gzip, deflate, br
connection: keep-alive
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelyMRKT/0.1.0
```

RESPONSE

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kR
wLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3
wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3
D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
```

```
content-encoding: br
alt-svc: h3=":8443"; ma=86400
<!DOCTYPE html>
<html lang="tr">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <title>test.anilyekta Giriş </title>
     <!-- Font Awesome -->
     <link rel="stylesheet" href="/Content/plugins/fontawesome-
free/css/all.min.css">
     <!-- Theme style -->
     <link rel="stylesheet" href="/Content/dist/css/adminlte.min.css">
</head>
<body class="hold-transition lockscreen dark-mode">
     <!-- Automatic element centering -->
     <div class="lockscreen-wrapper">
          <!-- START LOCK SCREEN ITEM -->
          <div class="lockscreen-item">
               <!-- lockscreen image -->
```

7 Weak cipher suites enabled

LOW

CVSS SCORE
4.2

CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N

PATH

https://test.anilyekta.com.tr:8443

DESCRIPTION

The server supports weak cipher suites for SSL/TLS connections. These cipher suites are currently considered broken and, dependin g on the specific cipher suite, offer poor or no security at all. Thus defeating the purpose of using a secure communication channel in the first place.

Any connection to the server using a weak cipher suite is at risk of being eavesdropped and tampered with by an attacker that ca n intercept connections. This is more likely to occur to Wi-Fi clients.

Depending on the cipher suites used, a connection may be at an immediate risk of being intercepted.

The following issues need to be addressed:

- CBC ciphers enabled. Potentially vulnerable to padding oracle attacks

EVIDENCE

The following weak ciphers are enabled:

- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

3 Referrer policy not defined

LOW

CVSS SCORE
3.1

CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

METHOD	PATH
GET	https://test.anilyekta.com.tr:8443

DESCRIPTION

The application does not prevent browsers from sending sensitive information to third party sites in the **referer** header.

Without a referrer policy, every time a user clicks a link that takes him to another origin (domain), the browser will add a **referer** header with the URL from which he is coming from. That URL may contain sensitive information, such as password recovery tokens or personal information, and it will be visible that other origin. For instance, if the user is at `example.com/password_recovery?unique_token=14f748d89d` and clicks a link to `example-analytics.com`, that origin will receive the complete password recovery URL in the headers and might be able to set the users password. The same happens for requests made automatically by the application, such as XHR ones.

Applications should set a secure referrer policy that prevents sensitive data from being sent to third party sites.

EVIDENCE

Response headers, missing the `Referrer-Policy` header:

```
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
```

REQUEST

```
GET / HTTP/2
host: test.anilyekta.com.tr:8443
accept: */*
accept-encoding: gzip, deflate, br
connection: keep-alive
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelyMRKT/0.1.0
```

RESPONSE

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
```

```
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
<!DOCTYPE html>
<html lang="tr">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <title>test.anilyekta Giriş </title>
     <!-- Font Awesome -->
     <link rel="stylesheet" href="/Content/plugins/fontawesome-
free/css/all.min.css">
     <!-- Theme style -->
     <link rel="stylesheet" href="/Content/dist/css/adminlte.min.css">
</head>
<body class="hold-transition lockscreen dark-mode">
     <!-- Automatic element centering -->
     <div class="lockscreen-wrapper">
          <!-- START LOCK SCREEN ITEM -->
          <div class="lockscreen-item">
               <!-- lockscreen image -->
```

5 Missing Content Security Policy header

LOW

CVSS SCORE
3.7

CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

METHOD	PATH
GET	https://test.anilyekta.com.tr:8443

DESCRIPTION

The Content Security Policy (CSP) is an HTTP header through which site owners define a set of security rules that the browser must follow when rendering their site. The most common usage is to define a list of approved sources of content that the browser can lo ad. This can be used to effectively mitigate Cross-Site Scripting (XSS) and Clickjacking attacks.

CSP is flexible enough for you to define from where the browser can load JavaScript, Stylesheets, images, or fonts, among other op tions. It can also be used in report mode only, a recommended approach before deploying strict rules in a live environment. Howev er, please note that report mode does not protect you, it just logs policy violations.

EVIDENCE

Response headers, missing the Content-Security-Policy header:

```
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7
R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc
90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
```

REQUEST

```
GET / HTTP/2
host: test.anilyekta.com.tr:8443
accept: */*
accept-encoding: gzip, deflate, br
connection: keep-alive
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelyMRKT/0.1.0
```

RESPONSE

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kR
wLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3
wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3
D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
```

```
content-encoding: br
alt-svc: h3=":8443"; ma=86400
<!DOCTYPE html>
<html lang="tr">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <title>test.anilyekta Giriş </title>
     <!-- Font Awesome -->
     <link rel="stylesheet" href="/Content/plugins/fontawesome-
free/css/all.min.css">
     <!-- Theme style -->
     <link rel="stylesheet" href="/Content/dist/css/adminlte.min.css">
</head>
<body class="hold-transition lockscreen dark-mode">
     <!-- Automatic element centering -->
     <div class="lockscreen-wrapper">
          <!-- START LOCK SCREEN ITEM -->
          <div class="lockscreen-item">
               <!-- lockscreen image -->
```

4 Missing clickjacking protection

LOW

CVSS SCORE
6.5

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N

METHOD	PATH
GET	https://test.anilyekta.com.tr:8443

DESCRIPTION

A **frameable response** occurs when one or multiple pages can be used on an iframe on any website. This allows the **clickjackin g** attack to be used.

Clickjacking is when an attacker a hidden iframe with multiple transparent or opaque layers above it, to trick a user into clicking on a button or link on the iframe when they were intending to click on the the top level page. Thus, the attacker is "hijacking" clic ks meant for the top level page and routing them to the iframe.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text bo xes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an inv isible frame controlled by the attacker.

EVIDENCE

Response headers, missing the `X-Frame-Options` header:

```
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7
R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc
90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
```

REQUEST

```
GET / HTTP/2
host: test.anilyekta.com.tr:8443
accept: */*
accept-encoding: gzip, deflate, br
connection: keep-alive
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelyMRKT/0.1.0
```

RESPONSE

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kR
wLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3
wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3
D%3D"}],"group":"cf-nel","max_age":604800}
```

```
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
<!DOCTYPE html>
<html lang="tr">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <title>test.anilyekta Giriş </title>
     <!-- Font Awesome -->
     <link rel="stylesheet" href="/Content/plugins/fontawesome-
free/css/all.min.css">
     <!-- Theme style -->
     <link rel="stylesheet" href="/Content/dist/css/adminlte.min.css">
</head>
<body class="hold-transition lockscreen dark-mode">
     <!-- Automatic element centering -->
     <div class="lockscreen-wrapper">
          <!-- START LOCK SCREEN ITEM -->
          <div class="lockscreen-item">
               <!-- lockscreen image -->
```

1 Browser content sniffing allowed

LOW

CVSS SCORE
4.7

CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:N

METHOD	PATH
GET	https://test.anilyekta.com.tr:8443

DESCRIPTION

The application allows browsers to try to mime-sniff the content-type of the responses. This means the browser may try to guess t he content-type by looking at the response content, and render it in way it was not intended to. This behavior may lead to the exe cution of malicious code, for instance, to explore an XSS vulnerability.

Applications should disable this behavior, forcing browsers to honor the content-type specified in the response. Without a specific c ontent-type set browsers will default to render the content as text, turning XSS payloads innocuous.

Disabling mime-sniffing should be seen as an extra layer of defense against XSS, and not as replacement of the recommended XSS prevention techniques.

EVIDENCE

Response headers, missing the X-Content-Type-Options header:

```
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kRwLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7
R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc
90uozjYamATg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
```

REQUEST

```
GET / HTTP/2
host: test.anilyekta.com.tr:8443
accept: */*
accept-encoding: gzip, deflate, br
connection: keep-alive
user-agent: Mozilla/5.0 (compatible; +https://probely.com/sos)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
ProbelyMRKT/0.1.0
```

RESPONSE

```
HTTP/2 200 OK
date: Sat, 14 Sep 2024 15:25:09 GMT
content-type: text/html; charset=utf-8
x-powered-by: ASP.NET
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=kR
wLq3%2Fky%2FLLhi4q2UjgRzXtikuv3C7R%2BCTSZZr%2FWk7oBPVZZ0DcE%2B%2FxWAcDTs8j4rsqM3
wnJ0SPdcnO4S%2B9kha4BTY2TwmFGfLgkzSyNWLCazibfMSLhMKhHzhe9BtkDN7rLc90uozjYamATg%3
D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
```

```
server: cloudflare
cf-ray: 8c3163588fc679e3-DUB
content-encoding: br
alt-svc: h3=":8443"; ma=86400
<!DOCTYPE html>
<html lang="tr">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <title>test.anilyekta Giriş </title>
     <!-- Font Awesome -->
     <link rel="stylesheet" href="/Content/plugins/fontawesome-
free/css/all.min.css">
     <!-- Theme style -->
     <link rel="stylesheet" href="/Content/dist/css/adminlte.min.css">
</head>
<body class="hold-transition lockscreen dark-mode">
     <!-- Automatic element centering -->
     <div class="lockscreen-wrapper">
          <!-- START LOCK SCREEN ITEM -->
          <div class="lockscreen-item">
               <!-- lockscreen image -->
```

Glossary

Term	Definition
Vulnerability	A type of security weakness that might occur in applications (e.g. Broken Authentication, Information Disclosure). Some vulnerabilities take their name not from the weakness itself, but from the attack that exploits it (e.g. SQL Injection, XSS, etc.).
Findings	An instance of a Vulnerability that was found in an application.

Severity Legend

To each finding is attributed a severity which sums up its overall risk

The severity is a compound metric that encompasses the likelihood of the finding being found and exploited by an attacker, the skill required to exploit it, and the impact of such exploitation. A finding that is easy to find, easy to exploit and the exploitation has high impact, will have a greater severity.

Different findings of the same type could have a different severity: we consider multiple factors to increase or decrease it, such as if the application has an authenticated area or not.

The following table describes the different severities:

Severity	Description	Examples
HIGH	These findings may have a direct impact in the application security, either clients or service owners, for instance by granting the attacker access to sensitive information.	SQL Injection OS Command Injection
MEDIUM	Medium findings usually don't have immediate impact alone, but combined with other findings may lead to a successful compromise of the application.	Cross-site Request Forgery Unencrypted Communications
LOW	Findings where either the exploit is not trivial, the impact is low, or the finding cannot be exploited by itself.	Directory Listing Clickjacking

Category Descriptions

The following pages contain descriptions of each vulnerability. For each vulnerability you will find a section explaining its impact, causes and prevention methods.
These descriptions are very generic, and whenever they are not enough to understand or fix a given finding, more information is provided for that finding in the Detailed Finding Descriptions section.

SQL Injection

Description

SQL Injections are the most common form of injections because SQL databases are very popular in dynamic web applications. This vulnerability allows an attacker to tamper existing SQL queries performed by the web application. Depending on the queries, the attacker might be able to access, modify or even destroy data from the database.

Since databases are commonly used to store private data, such as authentication information, personal user data and site content, if an attacker gains access to it, the consequences are typically very severe, ranging from defacement of the web application to users data leakage or loss, or even full control of the web application or database server.

Fix

To fix an SQL Injection you should use Prepared Statements. If an application exclusively uses prepared statements, the developer can be sure that no SQL injection will occur. Prepared Statements can be thought of as a kind of compiled template for the SQL that an application wants to run, that can be customized using variable parameters.

As an added bonus, if you're executing the same query several times, then it'll be even faster than when you're not using prepared statements. This is because when using prepared statements, the query needs to be parsed (prepared) only once, but can be executed multiple times with the same or different parameters.

SSL cookie without Secure flag

Description

The cookie secure flag is intended to prevent browsers from submitting the cookie in any HTTP requests that use an unencrypted connection, thus an attacker that is eavesdropping the connection will not be able to get that cookie.

A flag without the secure flag set will always be sent on every HTTP request that matches the scope of cookie, i.e. the domain for which it is set. What this means is that if your application inadvertently makes an HTTP request (without encryption), this request will carry the cookie and any attacker that can eavesdrop the victim traffic will be able to read that cookie.

If the cookie in question is the session cookie, the attacker will be able to hijack the victim account.

Fix

To fix a vulnerability of this type, you just need to set the Secure flag on the vulnerable cookie, effectively preventing it from being transmitted in unencrypted connections, i.e. over HTTP.

Depending on the language and technologies you are using, setting the Secure flag could mean to enable it or setting it to true, either on the code of the application itself or in a configuration file of the webserver or Content Management System (CMS) you are using.

HSTS header not enforced

Description

The application does not force users to connect over an encrypted channel, i.e. over HTTPS. If the user types the site address in the browser without starting with *https*, it will connect to it over an insecure channel, even if there is a redirect to HTTPS later. Even if the user types *https*, there may be links to the site in HTTP, forcing the user to navigate insecurely. An attacker that is able to intercept traffic between the victim

and the site or spoof the site's address can prevent the user from ever connecting to it over an encrypted channel. This way, the attacker is able to eavesdrop all communications between the victim and the server, including the victim's credentials, session cookie and other sensitive information.

Fix

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS).

You can do so by sending the `Strict-Transport-Security` header so that browsers will always enforce a secure connection to your site, regardless of the user typing *https* in the address.

An HSTS enabled server includes the following header in an HTTPS response: `Strict-Transport-Security: max-age=15768000;includeSubdomains` Please bear in mind that only HTTPS responses should have the HSTS header, because browsers ignore this header when sent over HTTP.

When the browser sees this, it will remember, for the given number of seconds, that the current domain should only be contacted over HTTPS. In the future, if the user types *http://* or omits the scheme, HTTPS is the default. In this example, which includes the option `includeSubdomains`, all requests to URLs in the current domain and subdomains will go over HTTPS. When you set `includeSubdomains` make sure you can serve all requests over HTTPS! It is, however, important that you add the option `includeSubdomains` whenever is possible.

Instead of changing your application, you should have the web server setting the header for you. If you are using Apache, just enable `mod_headers` and add the following line to your virtual host configuration: `Header always set Strict-Transport-Security "max-age=15768000;includeSubdomains"`

If you are using NGINX, just add this line to your host configuration: `add_header Strict-Transport-Security max-age=15768000;includeSubdomains`

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and may therefore be vulnerable to aforementioned SSL stripping attacks. To mitigate this risk, you can optionally ask the browser vendors to include your domain in a preloaded list, included in the browser, and afterwards add the 'preload' flag to the HSTS header.

Weak cipher suites enabled

Description

The server supports weak cipher suites for SSL/TLS connections. These cipher suites are currently considered broken and, depending on the specific cipher suite, offer poor or no security at all. Thus defeating the purpose of using a secure communication channel in the first place.

Any connection to the server using a weak cipher suite is at risk of being eavesdropped and tampered with by an attacker that can intercept connections. This is more likely to occur to Wi-Fi clients.

Depending on the cipher suites used, a connection may be at an immediate risk of being intercepted.

Fix

To stop using weak cipher suites, you must configure your web server cipher suite list accordingly.

Ideally, as a general guideline, you should remove any cipher suite containing references to NULL, anonymous, export, DES, 3DES, RC4, and MD5 algorithms. Additionally, remove any cipher suite containing ciphers with less than 128 bit security. You should also remove any CBC ciphers, as CBC ciphers may be vulnerable to padding oracle attacks.

You should enable ECDHE and GCM cipher suites to ensure proper security. Please note that these modern ciphers are available in newer versions of TLS only. You will need to enable TLSv1.2 and above (for GCM cipher suites).

To achieve this, we propose a modern cipher suite, based on these recommendations:

`TLS13-AES-256-GCM-SHA384:TLS13-CHACHA20-POLY1305-SHA256:TLS13-AES-128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256`

For most systems, changing TLS cipher suites, requires a change on the web server configuration file. Please refer to your web server documentation on how to do so.

Referrer policy not defined

Description

The application does not prevent browsers from sending sensitive information to third party sites in the **referer** header.

Without a referrer policy, every time a user clicks a link that takes him to another origin (domain), the browser will add a **referer** header with the URL from which he is coming from. That URL may contain sensitive information, such as password recovery tokens or personal information, and it will be visible that other origin. For instance, if the user is at `example.com/password_recovery?unique_token=14f748d89d` and clicks a link to `example-analytics.com`, that origin will receive the complete password recovery URL in the headers and might be able to set the users password. The same happens for requests made automatically by the application, such as XHR ones.

Applications should set a secure referrer policy that prevents sensitive data from being sent to third party sites.

Fix

This problem can be fixed by sending the header **Referrer-Policy** with a secure and valid value. There are different values available, but not all are considered secure. Please note that this header only supports one directive at a time. The following list explains each one and it is ordered from the safest to the least safe:

- `no-referrer`: never send the header.
- `same-origin`: send the full URL to requests to the same origin (exact scheme + domain)
- `strict-origin`: send only the domain part of the URL, but sends nothing when downgrading to HTTP.
- `origin`: similar to **strict-origin** without downgrade restriction.
- `strict-origin-when-cross-origin`: send full URL within the same origin, but only the domain part when sending to another origin. It sends nothing when downgrading to HTTP.
- `origin-when-cross-origin`: similar to **strict-origin-when-cross-origin** without the downgrade restriction.

Insecure options: * `no-referrer-when-downgrade`: sends the full URL when the scheme does not change. It will send if both origins are, for instance, HTTP. * `unsafe-url`: always sent the full URL

A possible, safe option is `strict-origin`, so the header would look like this:

```
Referrer-Policy: strict-origin
```

It is normally easy to enable the header in the web server configuration file, but it can also be done at the application level.

Please note that the referrer header is written `referer`, with a single `r` but the referrer policy header is properly written, with `rr`: `Referrer-Policy`.

Missing Content Security Policy header

Description

The Content Security Policy (CSP) is an HTTP header through which site owners define a set of security rules that the browser must follow when rendering their site. The most common usage is to define a list of approved sources of content that the browser can load. This can be used to effectively mitigate Cross-Site Scripting (XSS) and Clickjacking attacks.

CSP is flexible enough for you to define from where the browser can load JavaScript, Stylesheets, images, or fonts, among other options. It can also be used in report mode only, a recommended approach before deploying strict rules in a live environment. However, please note that report mode does not protect you, it just logs policy violations.

Fix

You can define a Content Security Policy by setting a header in your application. The header can look like this:

```
Content-Security-Policy: frame-ancestors 'none'; default-src 'self', script-src '*://*.example.com:*'
```

In this example, the **frame-ancestors** directive set to **'none'** indicates that the page cannot be placed inside a frame, not even by itself. The **default-src** defines the loading policy for all resources, in this case, they can be loaded from the current origin (protocol + domain + port). The example sets a more specific policy for scripts, through the **script-src**, restricting script loading to any subdomain of example.com.

The policy can be with different directives, and there are other less strict options for the directives above.

Missing clickjacking protection

Description

A **frameable response** occurs when one or multiple pages can be used on an iframe on any website. This allows the **clickjacking** attack to be used.

Clickjacking is when an attacker a hidden iframe with multiple transparent or opaque layers above it, to trick a user into clicking on a button or link on the iframe when they were intending to click on the the top level page. Thus, the attacker is "hijacking" clicks meant for the top level page and routing them to the iframe.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

Fix

The recommended way to prevent clickjacking is to send a header that instructs the browser to not allow arbitrary framing, typically from other domains.

The current recommendation is to use the Content-Security-Policy HTTP header (CSP) with a **frame-ancestors** directive. This header obsoletes the X-Frame-Options HTTP header.

To use CSP you need the following header:

```
Content-Security-Policy: frame-ancestors 'none'
```

The header might contain more directives, and there are other less strict options for the **frame-ancestors** directive.

If you want to use X-Frame-Options, send the proper HTTP header, with one of the following directives:

```
X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
```

A third directive, **ALLOW-FROM** is no longer supported by modern browsers.

If you specify **DENY**, all attempts to load the page in a frame will fail. **SAMEORIGIN** will allow the page to be loaded in the site including it in a frame is the same as the one serving the page.

The most common option is **DENY** when there is no need to load your pages on some other site.

To configure IIS to send the Content-Security-Policy header, add this your site's Web.config file:

```
<system.webServer>
  ...

  <httpProtocol>
    <customHeaders>
      <add name="Content-Security-Policy" value="frame-ancestors 'none'" />
    </customHeaders>
  </httpProtocol>

  ...
</system.webServer>
```

To configure IIS to send the X-Frame-Options header, add this your site's Web.config file:

```
<system.webServer>
  ...

  <httpProtocol>
    <customHeaders>
      <add name="X-Frame-Options" value="DENY" />
    </customHeaders>
  </httpProtocol>

  ...
</system.webServer>
```

Browser content sniffing allowed

Description

The application allows browsers to try to mime-sniff the content-type of the responses. This means the browser may try to guess the content-type by looking at the response content, and render it in way it was not intended to. This behavior may lead to the execution of malicious code, for instance, to explore an XSS vulnerability.

Applications should disable this behavior, forcing browsers to honor the content-type specified in the response. Without a specific content-type set browsers will default to render the content as text, turning XSS payloads innocuous.

Disabling mime-sniffing should be seen as an extra layer of defense against XSS, and not as replacement of the recommended XSS prevention techniques.

Fix

This problem can be fixed by sending the header **X-Content-Type-Options** with value **nosniff**, to force browsers to disable the content-type guessing (the sniffing).

The header should look this:

```
X-Content-Type-Options: nosniff
```

It is normally easy to enable the header in the web server configuration file, but it can also be done at application level.