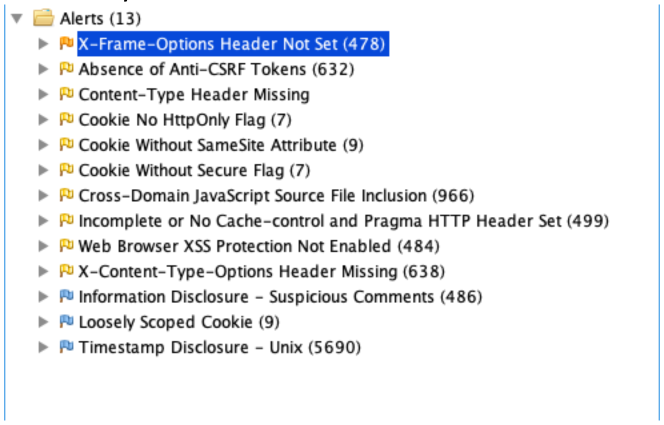
Phatcharapon Jumruspun 6031040921

Activity 9 – Web Security Scanner

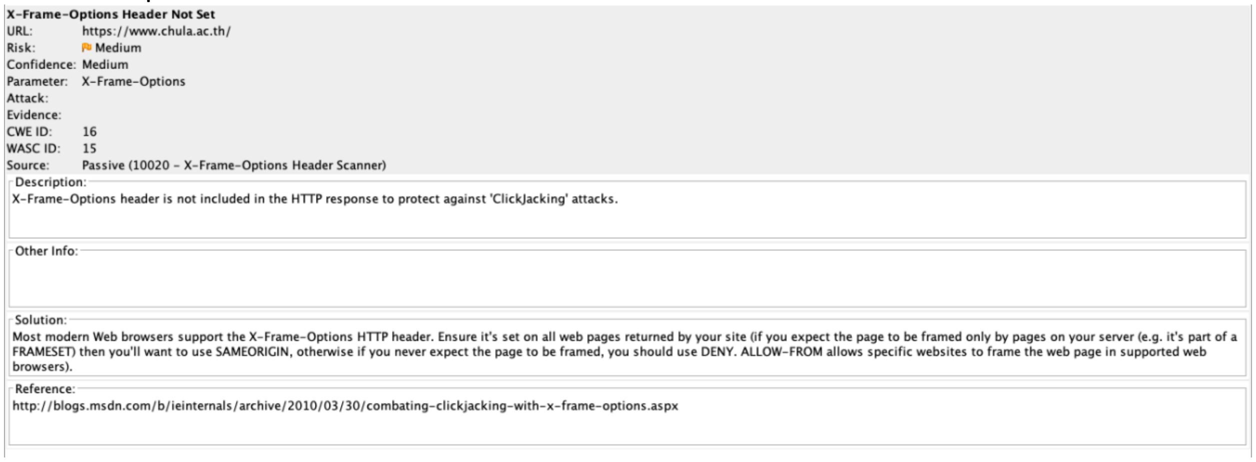
1) Overview of Vulnerability



```
Alerts (13)
  X-Frame-Options Header Not Set (478)
  Absence of Anti-CSRF Tokens (632)
  Content-Type Header Missing
  Cookie No HttpOnly Flag (7)
  Cookie Without SameSite Attribute (9)
  Cookie Without Secure Flag (7)
  Cross-Domain JavaScript Source File Inclusion (966)
  Incomplete or No Cache-control and Pragma HTTP Header Set (499)
  Web Browser XSS Protection Not Enabled (484)
  X-Content-Type-Options Header Missing (638)
  Information Disclosure - Suspicious Comments (486)
  Loosely Scoped Cookie (9)
  Timestamp Disclosure - Unix (5690)
```

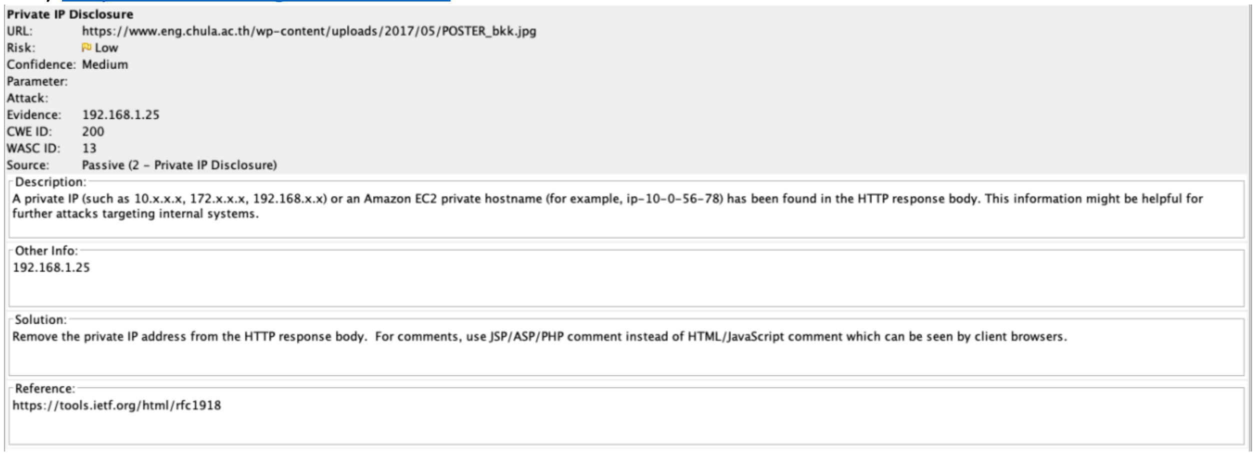
1.a) https://www.chula.ac.th/

X-Frame-Options Header Not Set



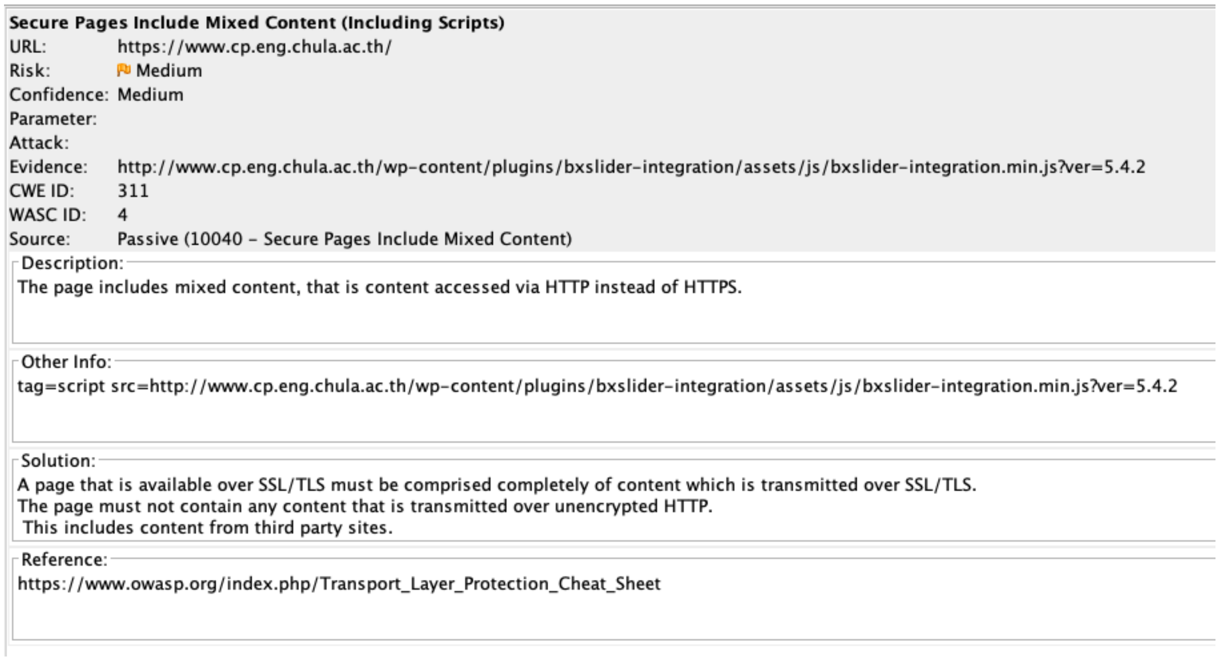
```
X-Frame-Options Header Not Set
URL:         https://www.chula.ac.th/
Risk:        Medium
Confidence:  Medium
Parameter:   X-Frame-Options
Attack:
Evidence:
CWE ID:      16
WASC ID:     15
Source:      Passive (10020 - X-Frame-Options Header Scanner)
Description:
X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Other Info:
Solution:
Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference:
http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx
```

1.b) https://www.eng.chula.ac.th/



```
Private IP Disclosure
URL:         https://www.eng.chula.ac.th/wp-content/uploads/2017/05/POSTER_bkk.jpg
Risk:        Low
Confidence:  Medium
Parameter:
Attack:
Evidence:    192.168.1.25
CWE ID:      200
WASC ID:     13
Source:      Passive (2 - Private IP Disclosure)
Description:
A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
Other Info:
192.168.1.25
Solution:
Remove the private IP address from the HTTP response body.  For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference:
https://tools.ietf.org/html/rfc1918
```

1.c) https://www.cp.eng.chula.ac.th/



Secure Pages Include Mixed Content (Including Scripts)

```
URL:        https://www.cp.eng.chula.ac.th/
Risk:       Medium
Confidence: Medium
Parameter:
Attack:
Evidence:   http://www.cp.eng.chula.ac.th/wp-content/plugins/bxslider-integration/assets/js/bxslider-integration.min.js?ver=5.4.2
CWE ID:     311
WASC ID:    4
Source:     Passive (10040 - Secure Pages Include Mixed Content)
```

Description:
The page includes mixed content, that is content accessed via HTTP instead of HTTPS.

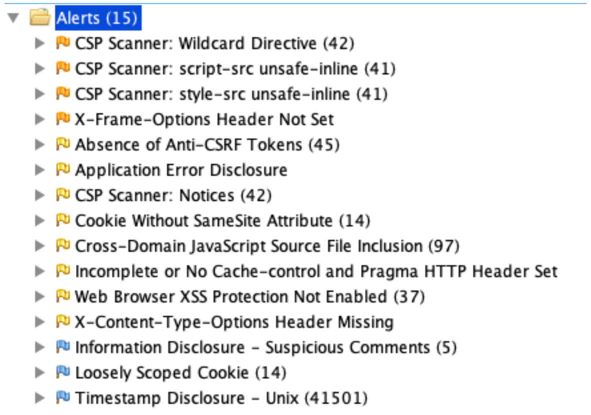
Other Info:
tag=script src=http://www.cp.eng.chula.ac.th/wp-content/plugins/bxslider-integration/assets/js/bxslider-integration.min.js?ver=5.4.2

Solution:
A page that is available over SSL/TLS must be comprised completely of content which is transmitted over SSL/TLS.
The page must not contain any content that is transmitted over unencrypted HTTP.
This includes content from third party sites.

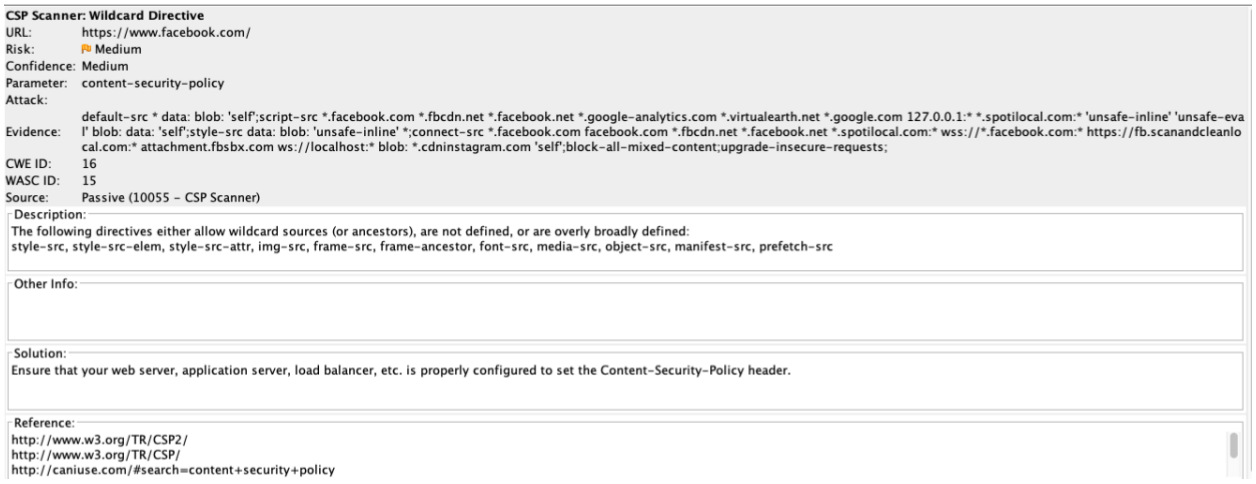
Reference:
https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

1.d) https://www.facebook.com/

Overviews



- Alerts (15)
  - CSP Scanner: Wildcard Directive (42)
  - CSP Scanner: script-src unsafe-inline (41)
  - CSP Scanner: style-src unsafe-inline (41)
  - X-Frame-Options Header Not Set
  - Absence of Anti-CSRF Tokens (45)
  - Application Error Disclosure
  - CSP Scanner: Notices (42)
  - Cookie Without SameSite Attribute (14)
  - Cross-Domain JavaScript Source File Inclusion (97)
  - Incomplete or No Cache-control and Pragma HTTP Header Set
  - Web Browser XSS Protection Not Enabled (37)
  - X-Content-Type-Options Header Missing
  - Information Disclosure - Suspicious Comments (5)
  - Loosely Scoped Cookie (14)
  - Timestamp Disclosure - Unix (41501)



CSP Scanner: Wildcard Directive

```
URL:        https://www.facebook.com/
Risk:       Medium
Confidence: Medium
Parameter:  content-security-policy
Attack:
Evidence:   default-src * data: blob: 'self';script-src *.facebook.com *.fbcdn.net *.facebook.net *.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' blob: data: 'self';style-src data: blob: 'unsafe-inline' *;connect-src *.facebook.com facebook.com *.fbcdn.net *.facebook.net *.spotilocal.com:* wss://*.facebook.com:* https://fb.scanandcleanlocal.com:* attachment.fbsbx.com ws://localhost:* blob: *.cdninstagram.com 'self';block-all-mixed-content;upgrade-insecure-requests;
CWE ID:     16
WASC ID:    15
Source:     Passive (10055 - CSP Scanner)
```

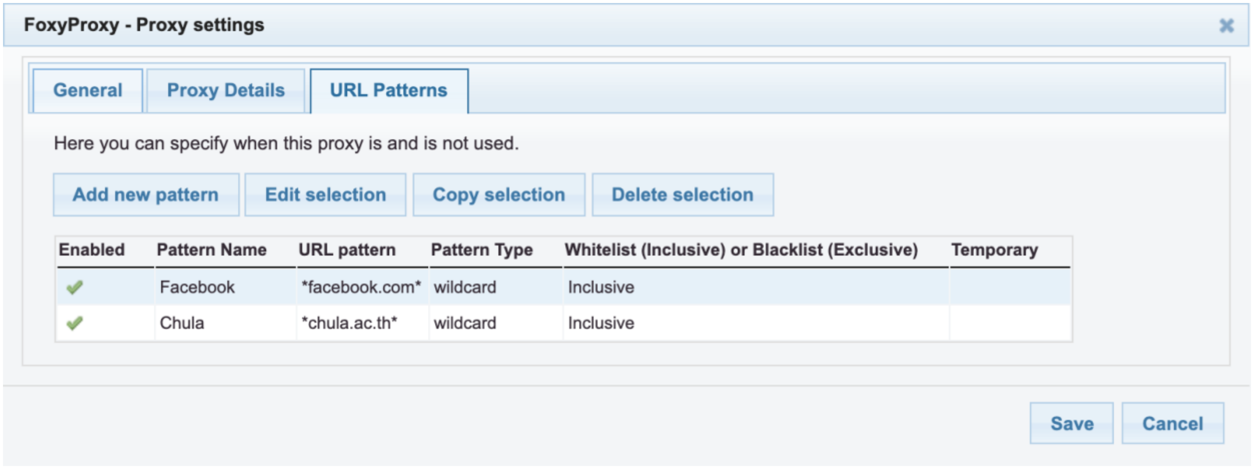
Description:
The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined:
style-src, style-src-elem, style-src-attr, img-src, frame-src, frame-ancestor, font-src, media-src, object-src, manifest-src, prefetch-src

Other Info:

Solution:
Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

Reference:
http://www.w3.org/TR/CSP2/
http://www.w3.org/TR/CSP/
http://caniuse.com/#search=content+security+policy

2) OWASP ZAP in proxy mode

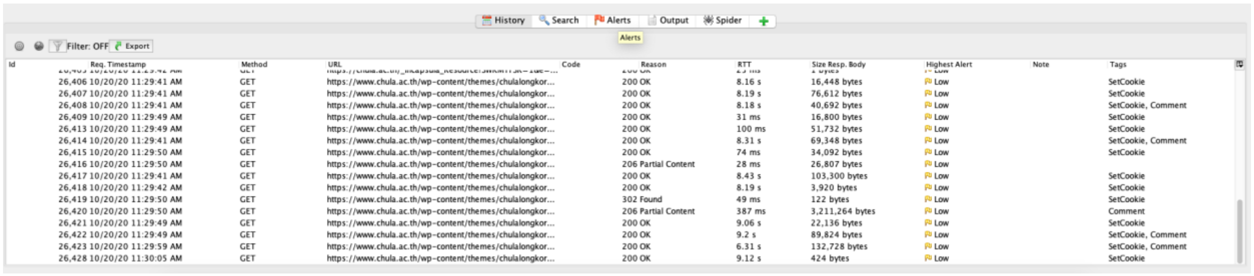


FoxyProxy - Proxy settings

Here you can specify when this proxy is and is not used.

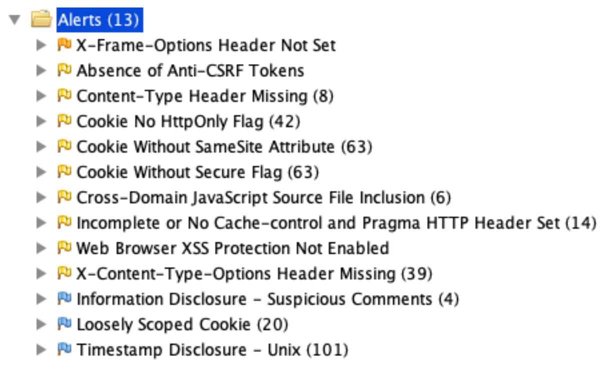
Enabled	Pattern Name	URL pattern	Pattern Type	Whitelist (Inclusive) or Blacklist (Exclusive)	Temporary
✓	Facebook	*facebook.com*	wildcard	Inclusive	
✓	Chula	*chula.ac.th*	wildcard	Inclusive	

FoxyProxy configuration page



Id	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
26,406	10/20/20 11:29:41 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.16 s	16,448 bytes	Low		SetCookie
26,407	10/20/20 11:29:41 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.19 s	76,612 bytes	Low		SetCookie
26,408	10/20/20 11:29:41 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.18 s	40,692 bytes	Low		SetCookie, Comment
26,409	10/20/20 11:29:49 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	31 ms	16,800 bytes	Low		SetCookie
26,413	10/20/20 11:29:49 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	100 ms	51,732 bytes	Low		SetCookie
26,414	10/20/20 11:29:41 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.31 s	69,348 bytes	Low		SetCookie, Comment
26,415	10/20/20 11:29:50 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	74 ms	34,092 bytes	Low		SetCookie
26,416	10/20/20 11:29:50 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		206 Partial Content	28 ms	26,807 bytes	Low		
26,417	10/20/20 11:29:41 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.43 s	103,300 bytes	Low		SetCookie
26,418	10/20/20 11:29:42 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	8.19 s	3,920 bytes	Low		SetCookie
26,419	10/20/20 11:29:50 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		302 Found	49 ms	122 bytes	Low		SetCookie
26,420	10/20/20 11:29:50 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		206 Partial Content	387 ms	3,211,264 bytes	Low		Comment
26,421	10/20/20 11:29:49 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	9.06 s	22,136 bytes	Low		SetCookie
26,422	10/20/20 11:29:49 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	9.2 s	89,824 bytes	Low		SetCookie, Comment
26,423	10/20/20 11:29:59 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	6.31 s	132,728 bytes	Low		SetCookie, Comment
26,428	10/20/20 11:30:05 AM	GET	https://www.chula.ac.th/wp-content/themes/chulalongkor...		200 OK	9.12 s	424 bytes	Low		SetCookie

Browsing history with details appeared when browsing in the browser

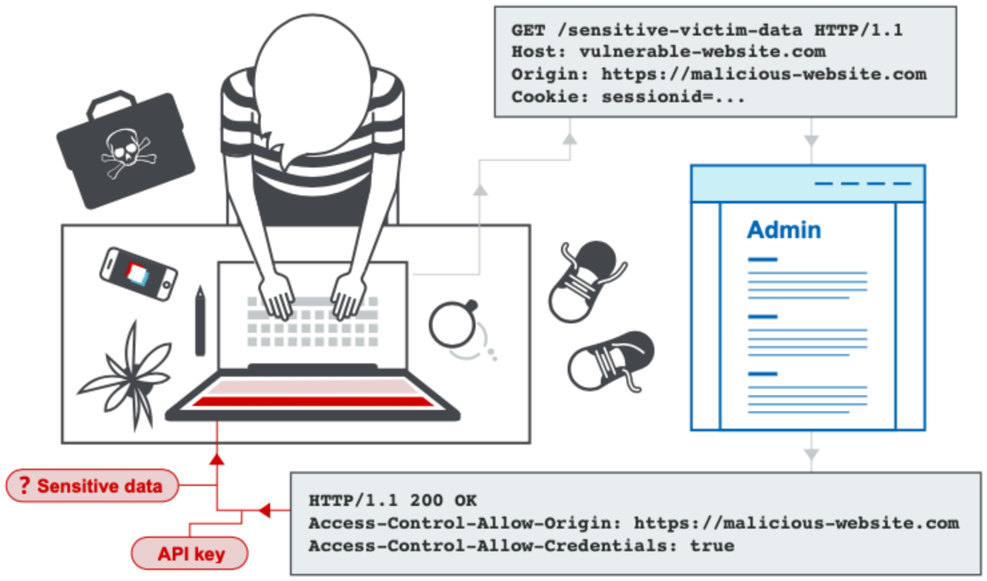


- Alerts (13)
  - X-Frame-Options Header Not Set
  - Absence of Anti-CSRF Tokens
  - Content-Type Header Missing (8)
  - Cookie No HttpOnly Flag (42)
  - Cookie Without SameSite Attribute (63)
  - Cookie Without Secure Flag (63)
  - Cross-Domain JavaScript Source File Inclusion (6)
  - Incomplete or No Cache-control and Pragma HTTP Header Set (14)
  - Web Browser XSS Protection Not Enabled
  - X-Content-Type-Options Header Missing (39)
  - Information Disclosure - Suspicious Comments (4)
  - Loosely Scoped Cookie (20)
  - Timestamp Disclosure - Unix (101)

Security Alerts also appears as we browse through the website

3) Vulnerability report in chula.ac.th domain name

3.1) Cross-Domain Misconfiguration



```
GET /sensitive-victim-data HTTP/1.1
Host: vulnerable-website.com
Origin: https://malicious-website.com
Cookie: sessionid=...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://malicious-website.com
Access-Control-Allow-Credentials: true
```

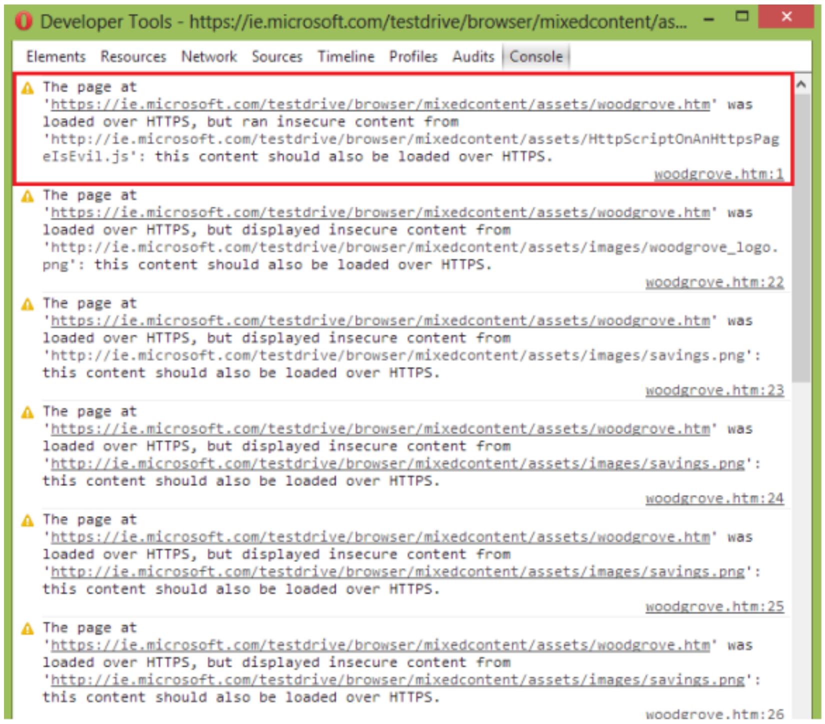
Risk: Medium
Confidence: Medium
Likelihood: Common (Due to easy misconfiguration)

Observation: https://www.chula.ac.th/calendar/index.php/calendar
Header Access-Control-Allow-Origin is set to * (asterisk)

Description: Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server

Recommendation:
- Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).
- Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

3.2) Secure Pages Include Mixed Content (Including Scripts)


Developer Tools - https://ie.microsoft.com/testdrive/browser/mixedcontent/as...
Elements Resources Network Sources Timeline Profiles Audits Console

```
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but ran insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/HttpScriptOnAnHttpsPag
eIsEvil.js': this content should also be loaded over HTTPS.
                                                                      woodgrove.htm:1
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but displayed insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/images/woodgrove_logo.
png': this content should also be loaded over HTTPS.
                                                                     woodgrove.htm:22
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but displayed insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/images/savings.png':
this content should also be loaded over HTTPS.
                                                                     woodgrove.htm:23
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but displayed insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/images/savings.png':
this content should also be loaded over HTTPS.
                                                                     woodgrove.htm:24
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but displayed insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/images/savings.png':
this content should also be loaded over HTTPS.
                                                                     woodgrove.htm:25
The page at
'https://ie.microsoft.com/testdrive/browser/mixedcontent/assets/woodgrove.htm' was
loaded over HTTPS, but displayed insecure content from
'http://ie.microsoft.com/testdrive/browser/mixedcontent/assets/images/savings.png':
this content should also be loaded over HTTPS.
                                                                     woodgrove.htm:26
```

HTTPS page loads some scripts through HTTP protocol

Risk: Medium (Data can leak in-between HTTP communication)
Confidence: Medium
Likelihood: Common
Evidence: http://www.chula.ac.th/calendar/images/calendar-banner.jpg

Observation:
These following contents are access through HTTP instead of HTTPS, which might result in security problems.

tag=img src=http://www.chula.ac.th/calendar/images/calendar-banner.jpg
tag=form action=http://www.chula.ac.th/calendar/index.php/calendar/search
tag=img src=http://www.chula.ac.th/calendar/images/feed-icon.png
tag=img src=http://www.chula.ac.th/calendar/images/us-icon.png
tag=img src=http://www.chula.ac.th/calendar/images/thai-icon.png
tag=script src=http://www.chula.ac.th/calendar/jquery/js/jquery-1.8.1.min.js
tag=script src=http://www.chula.ac.th/calendar/bootstrap/js/bootstrap.min.js
tag=script src=http://www.chula.ac.th/calendar/jquery/js/calendar.js
tag=script src=http://www.chula.ac.th/calendar/bootstrap/js/custom.js

Description: This occurs through HTTPS page loading scripts via HTTP protocol.

As we all know that HTTP protocol is unencrypted, thus is insecure, as the attacker can modify the content of the script being sent when they're being transported. With this, malicious scripts are loaded into the system. We can think of this as a man-in-the-middle attack.

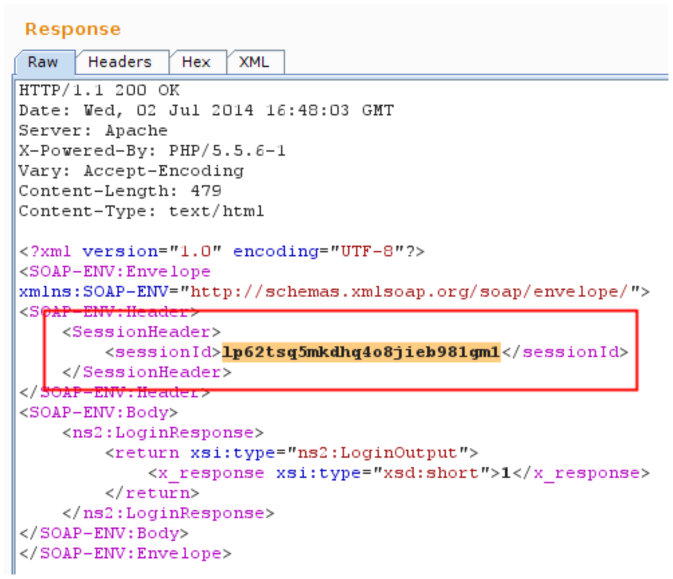
There are two types of Mixed-Content vulnerability. Passive and active.

- Mixed passive content: The page is loaded through HTTPS protocol, while static content that doesn't change the behaviors of the page (e.g. images) is loaded by HTTP. This is not as dangerous as mixed active content as the page's behaviors cannot be modified, only the images might face some unintended issues.

- Mixed active content: The page is loaded through HTTPS protocol, while scripts which can change the behavior of the page are loaded through HTTP. This can yield a greater damage as the attacker is allowed to change the website's behavior more freely.

Recommendation: None of the content should be loaded with unencrypted HTTP, use HTTPS instead.

3.3) Absence of Anti-CSRF Tokens



```
HTTP/1.1 200 OK
Date: Wed, 02 Jul 2014 16:48:03 GMT
Server: Apache
X-Powered-By: PHP/5.5.6-1
Vary: Accept-Encoding
Content-Length: 479
Content-Type: text/html

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
    <SessionHeader>
        <sessionId>1p62tsq5mkdhq4o8jieb981gm1</sessionId>
    </SessionHeader>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
    <ns2:LoginResponse>
        <return xsi:type="ns2:LoginOutput">
            <x_response xsi:type="xsd:short">1</x_response>
        </return>
    </ns2:LoginResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Image showing an Anti-CSRF token

Risk: Low
Confidence: Medium
Likelihood: Common

Observation: <form class="searchsite-form clearfix" role="search" method="get" action="https://www.chula.ac.th/" autocomplete="off">

Phatcharapon Jumruspun 6031040921

Description: No Anti-CSRF tokens were found in a HTML submission form.

The Cross-Site Request Forgery attack (CSRF/XSRF) is an attack when malicious requests are sent from an authenticated user to a web application. The attacker wouldn’t be able to see the response, thus stealing data is not possible. But the attacker gets to change the states of the application instead which can yield in a terrible damage to the system.

The attack is done by modifying a cross-site section of HTTP requests where the browser intended to service the users with content from another site. The attacker instead include their own malicious site in the value field, resulting in malicious data slipping into the overall system.

CSRF attack can easily be eliminated by just including tokens into the HTTP requests and make sure that the server will only process cross-site requests when a valid token is presented in the request. With this, even though the attacker tries to modify the HTTP request to have their malicious URL in it, they will unlikely be able to guess the valid token to perform any malicious actions.

Recommendation:
Phase: Architecture and Design

- There are libraries and packages you can use for preventing this attack to occur. For example, use anti-CSRF packages such as the OWASP CSRFGuard.
- Generate a unique nonce (e.g. token) for each HTTP request form, putting it in there, and verify the validity of the nonce before processing that particular request. The nonce should be almost impossible to guess. This can still be bypassed using Cross-site Scripting attacks.
- Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS.
- Use the ESAPI Session Management control. This control includes a component for CSRF.
- Do not use the GET method for any request that triggers a state change.

Phase: Implementation

- Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Implementation

- Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.