thinx-proxy Scan Report

Project Name	thinx-proxy
Scan Start	Friday, May 12, 2017 11:45:32 AM
Preset	Checkmarx Default
Scan Time	00h:01m:30s
Lines Of Code Scanned	2101
Files Scanned	21
Report Creation Time	Friday, May 12, 2017 11:49:06 AM
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000084&projectid=83
Team	CxServer
Checkmarx Version	8.4.1
Scan Type	Full
Source Origin	LocalPath
Density	1/1000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.1	All
OWASP Top 10 2013	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.1	None
OWASP Top 10 2013	None

Results Limit

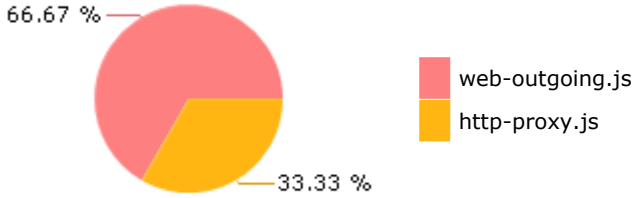
Results limit per query was set to 50

Selected Queries

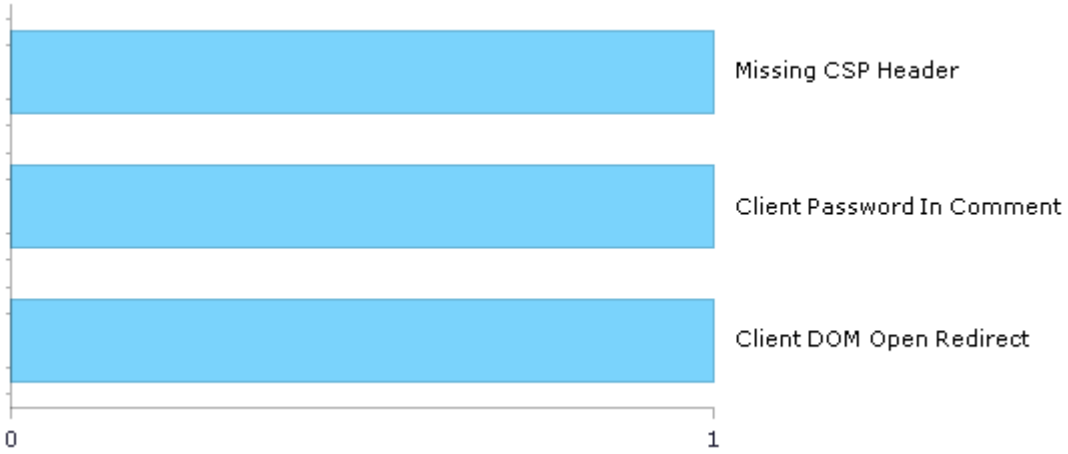
Selected queries are listed in Result Summary

Result Summary

Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References*	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	1	1
A7-Missing Function Level Access Control	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	1	1

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.1

Further details and elaboration about vulnerabilities and risks can be found at: PCI DSS v3.1

Category	Issues Found	Best Fix Locations
PCI DSS (3.1) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
PCI DSS (3.1) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.1) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.1) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.1) - 6.5.5 - Improper error handling	0	0
PCI DSS (3.1) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.1) - 6.5.8 - Improper access control	0	0
PCI DSS (3.1) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.1) - 6.5.10 - Broken authentication and session management	0	0

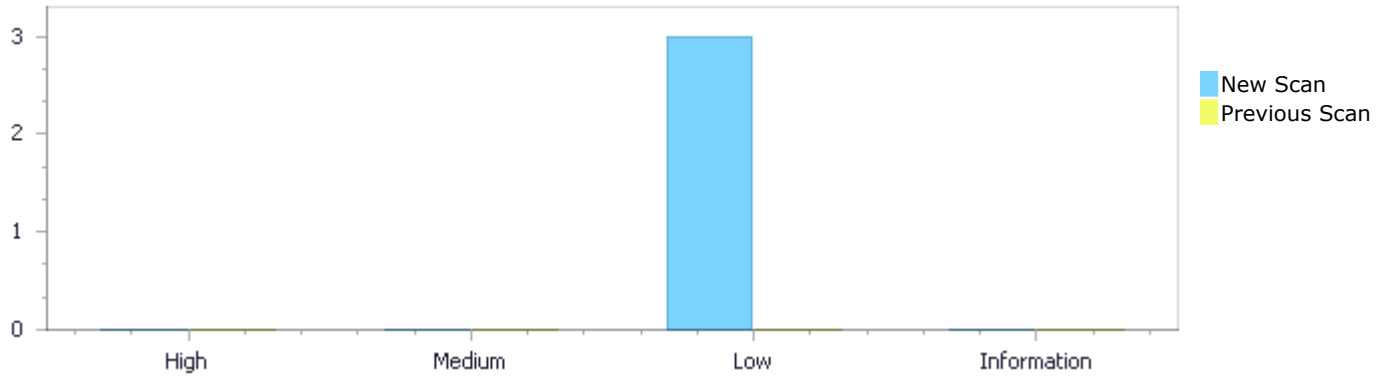
Scan Summary - Custom

Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	0	0	3	0	3
Recurrent Issues	0	0	0	0	0
Total	0	0	3	0	3
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	0	0	3	0	3
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	0	3	0	3

Result Summary

Vulnerability Type	Occurrences	Severity
Client DOM Open Redirect	1	Low
Client Password In Comment	1	Low
Missing CSP Header	1	Low

Scan Results Details

Client DOM Open Redirect

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client DOM Open Redirect Version:1

Categories

OWASP Top 10 2013: A10-Unvalidated Redirects and Forwards

Description

Client DOM Open Redirect\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000084&projectid=83&pathid=1
Status	New

A possible open redirect has been found at line 50 in /thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js file. This might lead to an untrusted site which mainly used for phising.

	Source	Destination
File	/thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js	/thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js
Line	55	71
Object	location	location

Code Snippet

File Name /thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js

Method setRedirectHostRewrite: function setRedirectHostRewrite(req, res, proxyRes, options) {

```
....
55.         var u = url.parse(proxyRes.headers['location']);
....
71.         proxyRes.headers['location'] = u.format();
```

Client Password In Comment

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Password In Comment Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure

Description

Client Password In Comment\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000084&projectid=83&

	pathid=2
Status	New

The application contains passwords embedded in source code comments, such as password at line 39 of /thinx-connect/node_modules/http-proxy/lib/http-proxy.js, which can easily be viewed by users.

	Source	Destination
File	/thinx-connect/node_modules/http-proxy/lib/http-proxy.js	/thinx-connect/node_modules/http-proxy/lib/http-proxy.js
Line	39	39
Object	password	password

Code Snippet
File Name /thinx-connect/node_modules/http-proxy/lib/http-proxy.js
Method * auth : Basic authentication i.e. 'user:password' to compute an Authorization header.

```
....
39.     *    auth    : Basic authentication i.e. 'user:password' to
compute an Authorization header.
```

Missing CSP Header

Query Path:
JavaScript\Cx\JavaScript Server Side Vulnerabilities\Missing CSP Header Version:1

Description

Missing CSP Header\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000084&projectid=83&pathid=3
Status	New

The application does not contain any code that sets Content Security Policy headers.

	Source	Destination
File	/thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js	/thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js
Line	94	94
Object	setHeader	setHeader

Code Snippet
File Name /thinx-connect/node_modules/http-proxy/lib/http-proxy/passes/web-outgoing.js
Method setHeader = function(key, header) {

```
....
94.             res.setHeader(String(key).trim(), header);
```

Client DOM Open Redirect

Risk

What might happen

An attacker could use social engineering to get a victim to click a link to the application, so that the user will be immediately redirected to another, arbitrary site. Users may think that they are still in the original application site. The second site may be offensive, contain malware, or, most commonly, be used for phishing.

Cause

How does it happen

The application redirects the user's browser to a URL provided in a user request, without warning users that they are being redirected outside the site. An attacker could use social engineering to get a victim to click a link to the application with a parameter defining another site to which the application will redirect the user's browser, and the user may not be aware of the redirection.

General Recommendations

How to avoid it

1. Ideally, do not allow arbitrary URLs for redirection. Instead, create a server-side mapping from user-provided parameter values to legitimate URLs.
2. If it is necessary to allow arbitrary URLs:
    - For URLs inside the application site, first filter and encode the user-provided parameter, and then use it as a relative URL by prefixing it with the application site domain.
    - For URLs outside the application (if necessary), use an intermediate disclaimer page to provide users with a clear warning that they are leaving your site.

Source Code Examples

CSharp

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Java

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Client Password In Comment

Risk

What might happen

It is often possible to retrieve and view the application source code. For web applications, it is even simpler to "`View Source`" in the user's browser. Thus, a malicious user can steal these passwords, and use them to impersonate whoever they belong to. It is not known if these are valid, current passwords, nor if they are user passwords or for backend systems, like a database.

Cause

How does it happen

A well-developed application will have it's source code well commented. Often, programmers will leave deployment information in comments, or retain debugging data that was used during development. These comments often contain secret data, such as passwords. These password comments are stored in the source code in perpetuity, and are not protected.

General Recommendations

How to avoid it

Do not store secrets, such as passwords, in source code comments.

Source Code Examples

JavaScript

Old code retained in comment with password

```
function login() {
    // send credentials to server instead of reading from database
    // constring = "Initial Catalog=mytest;User Id=sa;Pwd=mypass;";
    var creds = "username=" + txtUsername.text + "&password=" + txtPassword.text;

    var resp = sendToServer("/login", creds);
    return (resp == "success");
}
```

Passwords Cleaned from Comments

```
function login() {
    // send credentials to server instead of reading from database
    var creds = "username=" + txtUsername.text + "&password=" + txtPassword.text;

    var resp = sendToServer("/login", creds);
    return (resp == "success");
}
```

Missing CSP Header

Risk

What might happen

The Content Security Policy (CSP) HTTP header helps prevent unwanted content being injected into the application. This can help mitigate and prevent XSS vulnerabilities, unintended tracking, unacceptable frames, and other malicious content, and constrain embedded scripts, according to a whitelist. Without this policy, an attacker may be able to inject many types of malicious content (including scripts), without any constraint whatsoever.

Cause

How does it happen

The Content Security Policy (CSP) HTTP header must be explicitly set on the HTTP response, and the policy must be properly tuned to the application's requirements. This header was not set neither for the global application, nor for any specific request.

General Recommendations

How to avoid it

- Explicitly set the Content Security Policy (CSP) HTTP header with an appropriate policy.
- Create and set the CSP header to the response directly, or use the "helmet" package.
- Ensure the policy is finely tuned and matches the application's requirements.

Source Code Examples

JavaScript

Setting a CSP header with Helmet

```
var express = require('express');
var helmet = require('helmet');

var app = express();

app.use(helmet.contentSecurityPolicy({
    script-src: ["'self'", 'https://apis.google.com'],
    default-src: ['https://cdn.example.net'],
    child-src: ['https://content.example.net'],
    object-src: ["'none'"]
}));
```

Scanned Languages

Language	Hash Number	Change Date
JavaScript	0134728271507297	3/9/2017