OutSystems が実装するセキュリティ機能

情報技術開発株式会社

今や人々の生活の一部となっているインターネットですが、利便性向上の一方で個人情報の漏洩や技術の悪用による盗聴、企業・団体のホームページの改ざんなど、情報セキュリティインシデントに関するニュースを聞かない日はないほど情報セキュリティへの対策の重要性が増しています。

また急速に変化するビジネス環境に対して、迅速かつ柔軟に対応する必要が企業には求められており、DX（デジタルトランスフォーメーション）推進は必要不可欠な状況であり、その DX を推進する手段として注目されているローコード/ノーコード開発が注目を集めています。

ローコード開発プラットフォーム製品の一つである OutSystems で構築したアプリケーションはセキュリティへの対策がどの程度のものなのか、本書では OutSystems が実装するセキュリティ機能についてほんの一部ではありますが、ご紹介いたします。

内容

脆弱性対策への重要性

IPA が公表している「2023 年第 3 四半期に登録された脆弱性の種類別件数」※1 によると、クロスサイトスクリプティング（登録件数：1,896 件）を筆頭に、SQL インジェクション（同：1,153 件）、境界外書き込み（同：1,036 件）となっており、アプリケーションの脆弱性に関するインシデントはもはや日常化し、Web アプリの開発者は、アプリケーションやソフトウェアの企画・設計段階から、さまざまな脆弱性への対応が求められています。

※1 出典：https://www.ipa.go.jp/security/reports/vuln/jvn/ipedia2023q3.html

アプリケーションやソフトウェアに存在する脆弱性を放置すると、悪意を持った第三者はその脆弱性を足がかりに、システムの乗っ取りやマルウェアを利用して企業の機密情報を不正に入手するなどの攻撃を行います。脆弱性を放置したことにより、企業の社会的信用の失墜、情報漏洩による賠償金の支払いや事業の停止、最悪の場合は廃業せざるを得なくなるなど深刻な状況を招く恐れがあります。

では、OutSystems を使用して構築した Web アプリは脆弱性に対応しているのか、本当にセキュアに構築できるのか、その対策方法を見ていきます。

OutSystems が提供するセキュリティ機能

最初に OutSystems がデフォルトで提供するセキュリティ対策機能※2 を確認します。

1. アプリケーションコードの保護
   - OutSystem に組み込まれたセキュアなコードパターンにより脆弱性からアプリケーションを保護
2. セッションデータの保護
   - ログインごとにセッション ID を透過的に変更、リクエストごとに検証を行うことでセッション固定攻撃から保護
3. 認証メカニズムの保護
   - 認証 Cookie の認証情報を暗号化・保持する認証メカニズムを組み込み
   - SSO（（Single-Sign-On）への対応
4. アプリケーションのロールベースのアクセス制御
   - 特定のロールに基づいたアプリケーションのページへのアクセス制限
   - 視覚的構成要素によるロール権限の定義
   - 特定のビジネス機能の実行可否制御
5. IT ユーザーのロールベースのアクセス制御
   - ロールに基づいた IT チームの役割定義
   - 割り当てられたロールによるアプリケーションの本番移行可否制御など

6. 総当たり攻撃からの保護
   - エンドユーザーと IT ユーザーを総当たり攻撃から保護するメカニズムを組み込み
7. HTTPS の適用
   - エンドポイント間の通信を HTTPS 経由とすることにより通信をすべて暗号化
8. アプリケーションの監査
   - 組み込み済み監視ツールによる実行中アプリのデータ収集・照会
   - アプリのログ・エラー、画面リクエスト、ビジネスプロセス、セキュリティ監査など特定の環境ログおよび状態の照会が可能
9. システムアクティビティの監査
   - IT ユーザーの実行したタスクの監視
   - デプロイ、ユーザー構成の変更、システムのログインなど
10. 脆弱性の管理
    - 製品および生成されるコードに含まれる脆弱性の継続的に監視
    - OutSystems インフラのアップデートによる最新機能・修正版の利用

※2 出典：https://success.outsystems.com/ja-jp/support/security/develop_secure_outsystems_apps/

このように、OutSystems のセキュリティ対策は十分に実装されていますが、アプリケーションの構築方法次第では、脆弱性を生んでしまうこともあります。本書ではその一部のケースと、対応方法を解説していきます。

今回取り上げるセキュリティ対策の内容は以下の 3 点です。

- インジェクション攻撃への対応
- CSP（コンテンツ・セキュリティ・ポリシー）
- Cookie セキュア属性

1. インジェクション攻撃への対応

インジェクション攻撃とは

インジェクションは直訳すると、「注射」「注入」「噴射」という意味です。IT 用語では、悪意を持った利用者が Web の入力画面から、特定の文字列を入力して不正にデータを取得したり、スクリプトを Web サーバーに送り込んでマルウェアをインストールしたり、コードを実行することを指します。

インジェクションが起こりうる原因は、構築した Web アプリケーションの画面において、利用者が入力した値を適切にエスケープ処理（プログラムで使用する特殊な記号をただの文字として処理すること）を行わないことが挙げられます。

エスケープ処理を実施していない Web アプリの動作がどのようなものか、SQL インジェクションを例に説明します。「図 1 SQL インジェクション例」にあるように、悪意のある利用者が、Web アプリの入力画面から不正なスクリプトを入力し（①、②）AP/DB サーバー側で受け取ったスクリプトを処理（③）してしまい、本来入手されることのないデータや情報が、悪意のある利用者にわたってしまいます（④、⑤）。

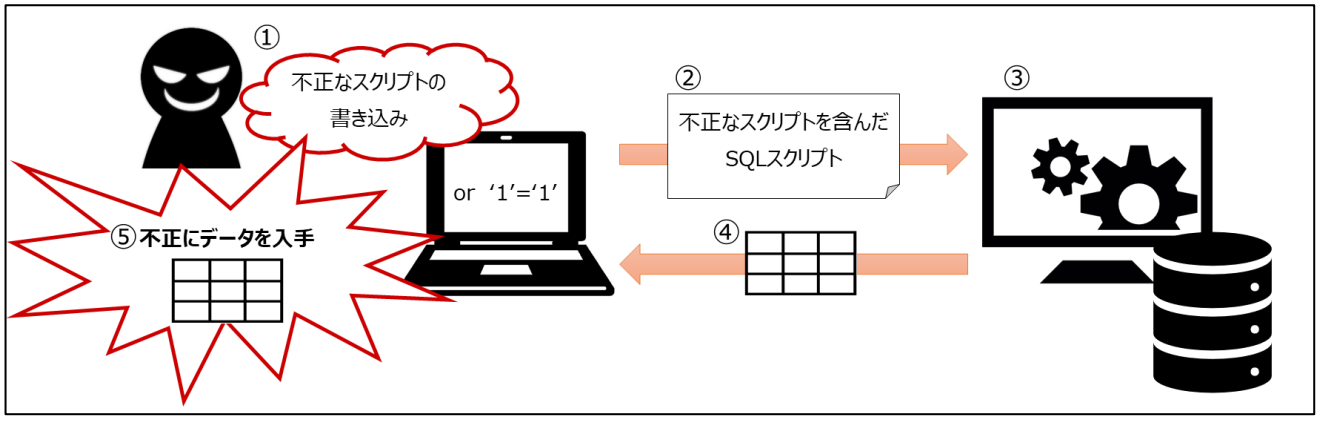

図 1 SQL インジェクション例

OutSystems では、コード生成時に自動でセキュアなコードパターンを組み込み、.NET コードを自動生成し、インジェクション攻撃のみならず XSS（クロス・サイト・スクリプティング）など他の脆弱性からも保護されたアプリを構築する仕組みを実装しています。

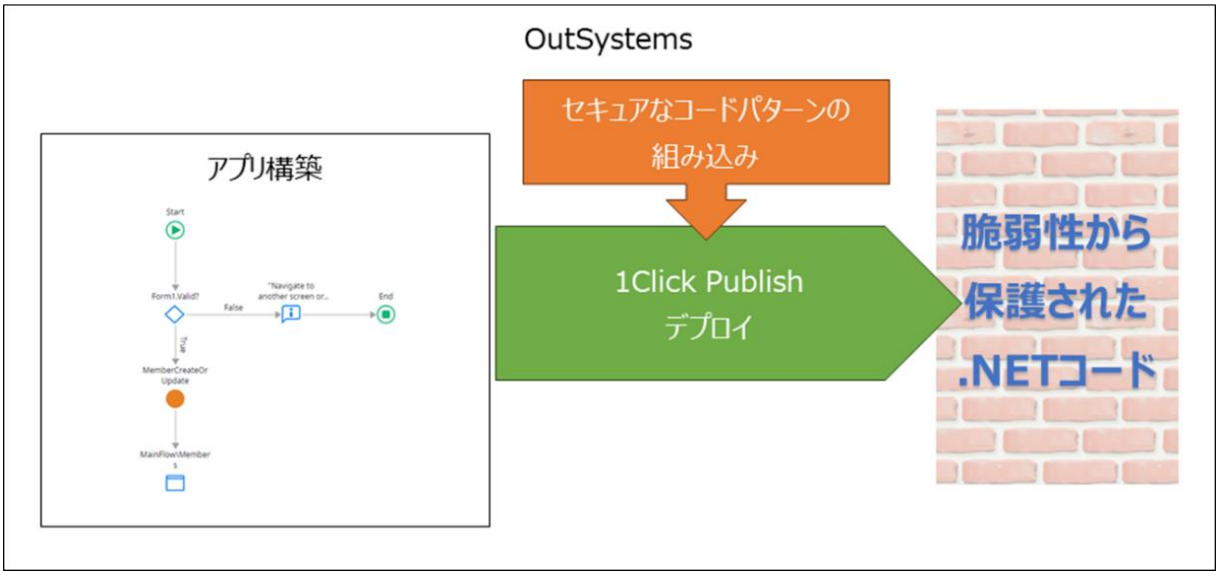

図 2 OutSystems のコードパターン生成イメージ

SQL インジェクション回避の実装例

OutSystems で構築したアプリでもインジェクション攻撃の対応が必要になるケースがあります。SQL インジェクションを例にすると、以下の条件に当てはまる場合です。

- データ取得条件が Web アプリ画面の Input タグを通してパラメータとして引き渡される
- AdvancedSQL を使用し、動的にデータ取得条件を変更する必要がある

具体的に言うと、「図 3 SQL ステートメント例」のような SQL を発行する必要がある場合です。

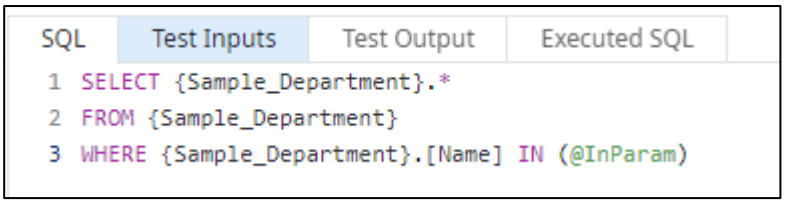

図 3 SQL ステートメント例

この SQL ステートメントは、データ取得条件に IN 句を使用しており、IN 句の値がパラメータ値@InParam により動的に変わります。このときパラメータ値@InParam はエスケープ処理されている必要がありますが、デフォルトではエスケープ処理はされません。

実際、SQL インジェクションの危険性があるアクションが作成された場合、OutSystems は True Change に「図 4 True Change メッセージ 1」に示したような SQL インジェクションの脆弱性を指摘するメッセージをワーニングとして表示します。

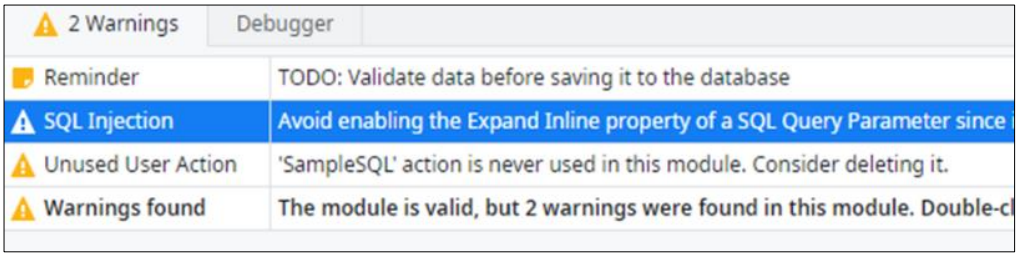

図 4 True Change メッセージ 1

これを回避するためには AdvancedSQL に引き渡すパラメータをエスケープ処理する必要があります。文字列をエスケープ処理するためには、OutSystems が提供している Sanitization（extension）を使用します。

まず、「Management Dependencies」ダイアログ（メニューバーで選択、またはコンセントマークのアイコン）を開き、Sanitization を探し選択します（図 5 Management Dependencies）。

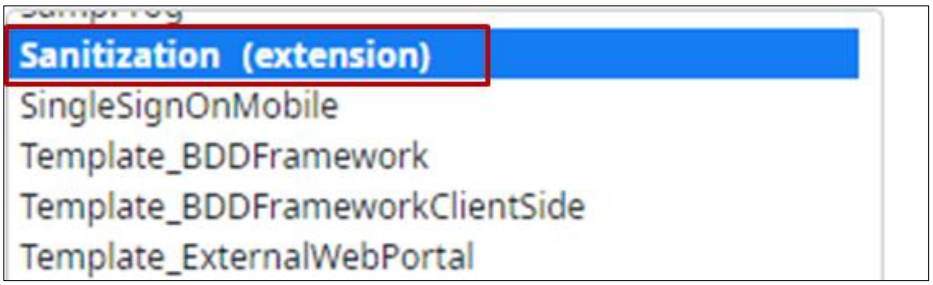

図 5 Management Dependencies

Sanitization（extension）を選択すると、ダイアログの右側に Sanitization に定義されているエレメントが「図 6 Sanitization エレメント」のように表示されますので、そこから BuildSafe_InclauseTextList（数値項目をエスケープする場合は、～IntegerList）を選択します。Structure は、該当利用するサーバーアクションを選択すると自動で選択されます。

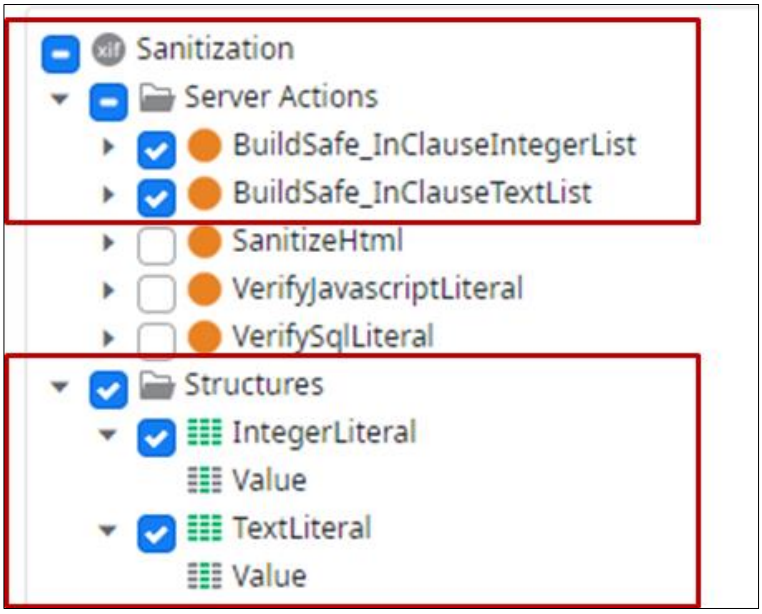

図 6 Sanitization エレメント

Sanitization の参照設定ができたら次はロジックを作成します。実際のロジックを「図 7 実装例」に示します。

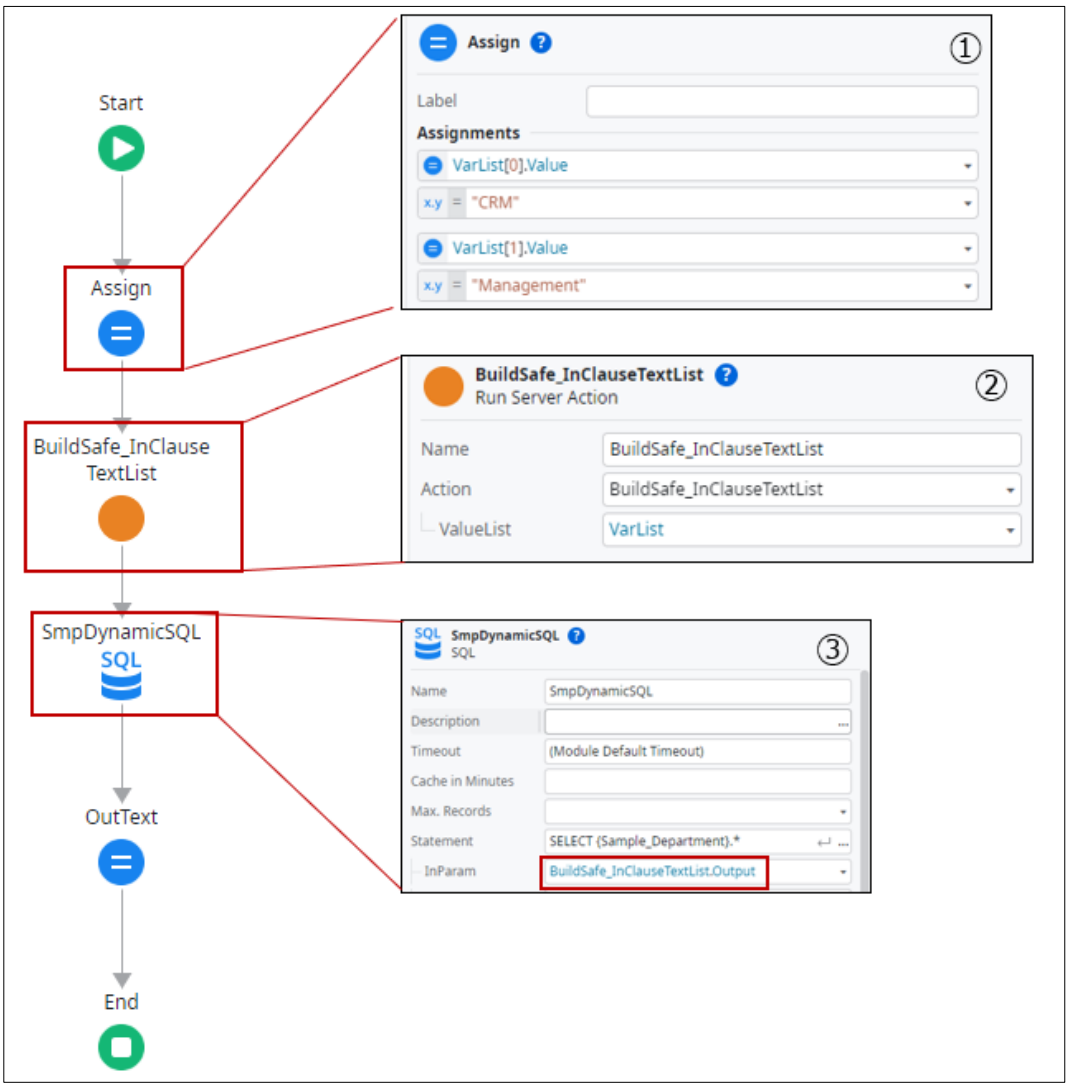

図 7 実装例

① BuildSafe_InClauseTextList に引き渡す List 型変数（VarList）に値をセット

※通常 ForEach を使用してループ処理で行いますが、実装例では Assign ウィジェットで指数を使用し List 型変数に値をセットする形としています。

② BuildSafe_InClauseTextList を配置し、条件とする文字列をエスケープ処理

※引数は①の Assign でセットした VarList

③ AdvancedSQL を配置し、データ取得の SQL（図 3 SQL ステートメント例）を記述

※AdvancedSQL の引数は BuildSafe_InClauseTextList の戻り値

このように文字列を適切にエスケープ処理すると、「図 8 True Change メッセージ 2」で確認できるように SQL インジェクションのワーニングメッセージが表示されなくなり、脆弱性を回避できたということが言えます。

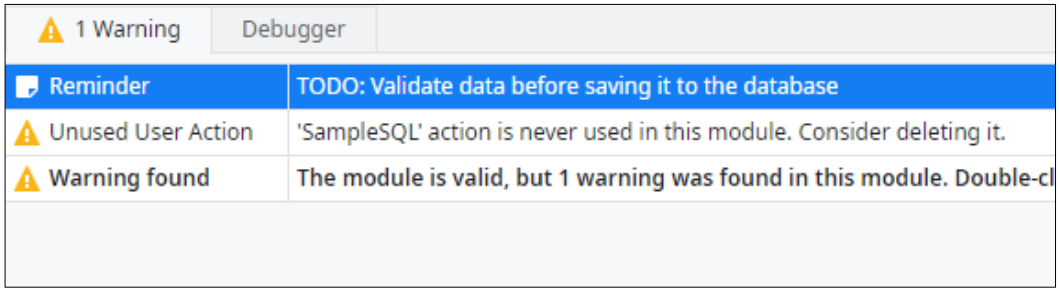

図 8 True Change メッセージ 2

2. コンテンツ・セキュリティ・ポリシー（CSP）の設定

コンテンツ・セキュリティ・ポリシー（CSP）とは

コンテンツ・セキュリティ・ポリシー（以下 CSP）は、XXS、データインジェクション攻撃などの特定の攻撃の検出と軽減を支援するセキュリティレイヤの一つです。CSP を設定すると、Web サーバーがブラウザにリクエストを返す際に HTTP レスポンスに Content-Security-Policy ヘッダが設定されます。Content-Security-Policy ヘッダは、コンテンツの提供元や取得方法が記載されており、コンテンツ提供者が意図しないコンテンツをブラウザが読み込むことがないようにすることができます。

OutSystems における CSP の設定

OutSystems ではこの CSP の設定を LifeTime にて、環境（開発/テスト/本番）またはアプリケーションの単位で行うことができますが、LifeTime を導入していない場合は Service Center で行うことができます。

CSP の設定は、LifeTime のメニューバーにある「インフラストラクチャ」（英語表記されている場合は、「Infrastructure」） から対象環境の「環境のセキュリティ」（同「Environment Security」）を選択し環境のセキュリティ設定画面（図 9 OutSystems の CSP 設定画面）で行います。各項目に対してそれが何のための項目であるのか説明書きがありますので、わかりやすくなっています。

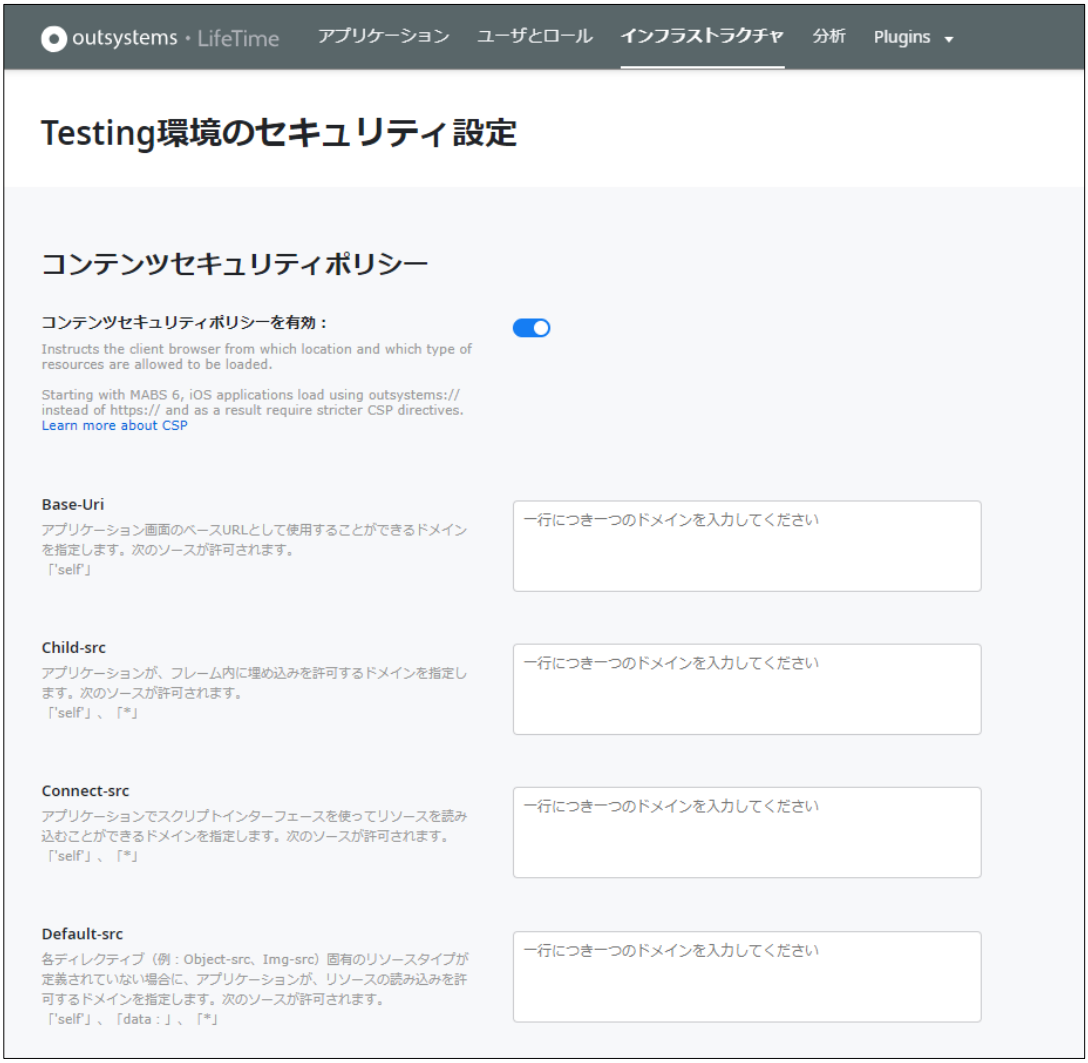

図 9 OutSystems の CSP 設定画面

また保存ボタン付近には、キーワードの説明があり、利用者がキーワードの意味を理解できるようになっています。

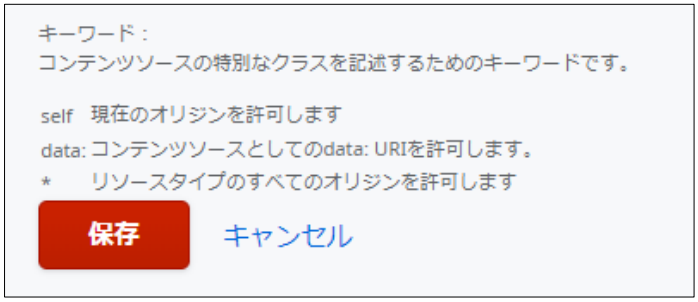

図 10 CSP 項目の保存ボタン付近

CSP の設定と Web ページの確認

実際に CSP 項目を入力してみます。「図 11 CSP 項目に値を入力」にあるように、Connect-src と Default-src に'self'と入力し保存ボタンを押します。

図 11 CSP 項目に値を入力

保存ボタンを押すと、図 12 のようなメッセージが表示されます。

これは OutSystems が CSP の値を保存したことに加えて設定した内容を検証し、入力した内容により不整合が起こる可能性がある場合、OutSystems が判断し他の CSP 項目を自動でセットしました、という内容になります。したがって OutSystems における CSP の設定は注意を払うべきコンテンツ管理を行えばよく、その他デフォルト値で対応可能な項目については空白のままで設定すると OutSystems が補完してくれる、ということが言えます。

図 12 保存ボタン押下時のメッセージ

実際に OutSystems が自動で CSP 項目をセットした画面が「図 13 CSP 項目が自動セットされた CSP 画面」です。

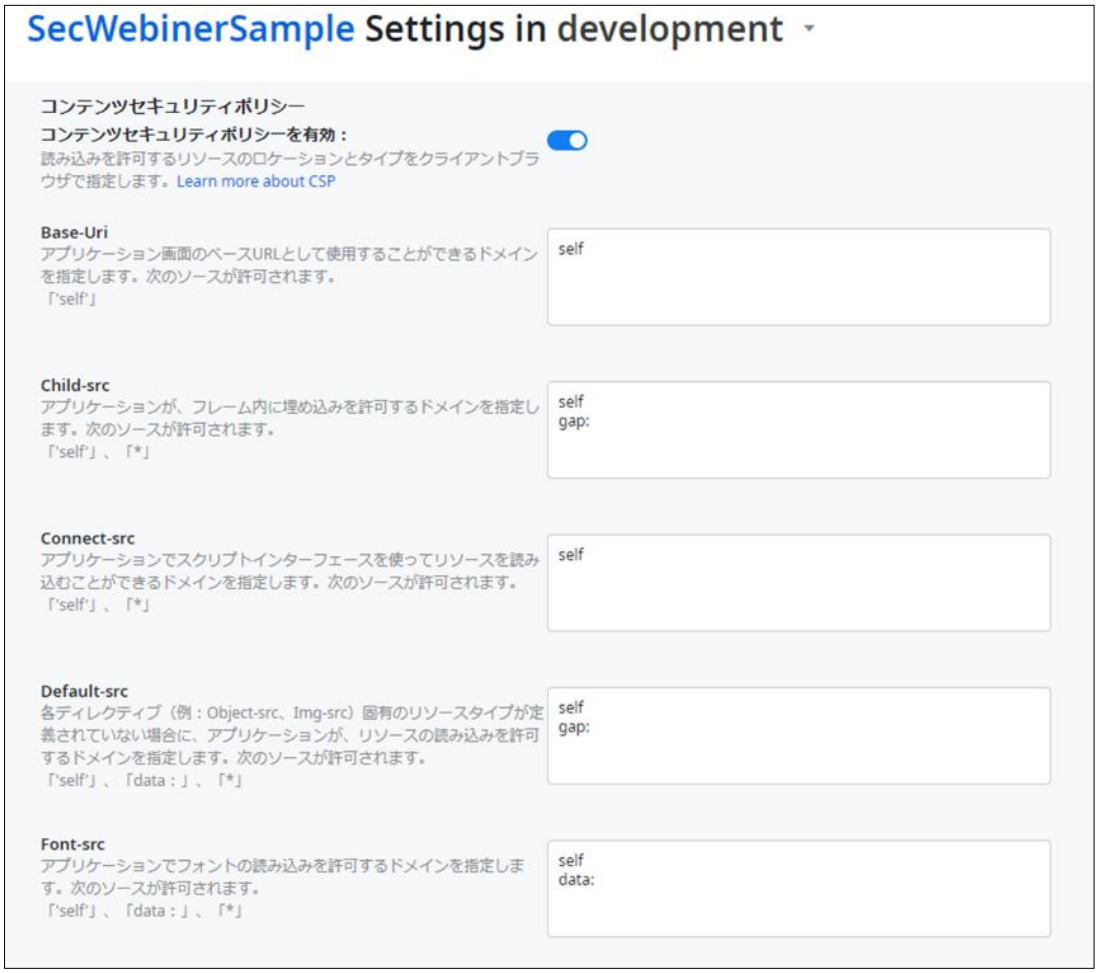

図 13 CSP 項目が自動セットされた CSP 画面

実際に CSP を設定したアプリケーションを実行し、HTTP レスポンスがどのようになるか確認します。HTTP ヘッダは Chrome のデベロッパーツールで確認します。

CSP設定前

```
HTTP/1.1 200 OK
Date: Thu, 14 Dec 2023 05:09:31 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: no-cache
Pragma: no-cache
Content-Encoding: gzip
Expires: -1
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
```

CSP設定後

```
HTTP/1.1 200 OK
Date: Thu, 14 Dec 2023 05:16:55 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: no-cache
Pragma: no-cache
Content-Encoding: gzip
Expires: -1
Vary: Accept-Encoding
Content-Security-Policy: base-uri 'self'; child-src 'self' gap:; frame-src 'self' gap:;
X-Content-Security-Policy: base-uri 'self'; child-src 'self' gap:; frame-src 'self' ga
X-WebKit-CSP: base-uri 'self'; child-src 'self' gap:; frame-src 'self' gap:; connect-
X-Content-Type-Options: nosniff
```

図 12 HTTP ヘッダの内容

図 12 が HTTP レスポンスの内容です。左が CSP 設定前、右が CSP 設定後に実行したものになります。CSP 設定後の HTTP レスポンスは、Content-Security-Policy ヘッダが追加され CSP で設定した内容が含まれていることがわかります。

このように OutSystems では簡単に CSP の設定を行えることがお判りいただけたと思います。OutSystems で設定可能な CSP 設定項目を表 1 にまとめていますので、ご参考いただければと思います。

ディレクティブ	説明
Base-uri	アプリケーション画面のベースURLとして使用できるドメイン
Child-src	アプリケーションがフレームの組み込みを許可されているドメイン
Connect-src	アプリケーションがスクリプトインターフェイスを使用してリソースを読み込むことが許可されているドメイン
Default-src	アプリケーションがデフォルトでリソースの読み込みを許可されているドメイン
Font-src	アプリケーションがフォントの読み込みを許可されているドメイン
Img-src	アプリケーションが画像の読み込みを許可されているドメイン
Media-src	アプリケーションがメディアファイルの読み込みを許可されているドメイン
Object-src	アプリケーションがオブジェクト（<object>、<embed>、<applet>要素用）の読み込みを許可されているドメイン
Plugin-types	ユーザーのブラウザが呼び出せる有効なプラグイン
Script-src	アプリケーションがスクリプトの読み込みを許可されているドメイン
Style-src	アプリケーションがスタイルの読み込みを許可されているドメイン
Frame-ancestors	フレームにアプリケーションを組み込むことが許可されているドメイン
Frame-src	専用のフィールドはありませんが、［Child-src］フィールドを使用してプラットフォームの値を入力すると、Frame-srcディレクティブの生成が可能
Report-to	コンテンツセキュリティ違反が報告されるURI
Other directives	CSPのヘッダーに付加されるその他のディレクティブ

表 1 OutSystems における CSP 設定項目一覧

出典：https://success.outsystems.com/ja-jp/documentation/11/managing_the_applications_lifecycle/secure_the_applications/apply_content_security_policy/

3. Cookie のセキュア属性の設定

Cookie のセキュア属性とは

Cookie は、広く知られている通り Web サーバーからブラウザに送信される制御情報の一つで、主にはセッション管理（例：ログイン情報）、パーソナライズ（例：ユーザー設定情報）、トラッキング（例：ユーザー行動の記録や分析）の用途で使用されます。

Cookie は、通信方法が HTTP・HTTPS にかかわらず Web サーバーからクライアント側のブラウザに送信されますが、HTTP 通信の場合、容易に盗聴され、例えばセッションハイジャックなどの攻撃により不正に悪用されるリスクが高くなります。また仮に HTTPS 通信が適用されるサイトを利用していたとしても、誤って HTTP 通信を行う悪意のあるサイトのリンクを踏んでしまった場合、Cookie を盗聴されてしまうことがあります。このようなことを防ぐために Cookie のセキュア属性を設定します。

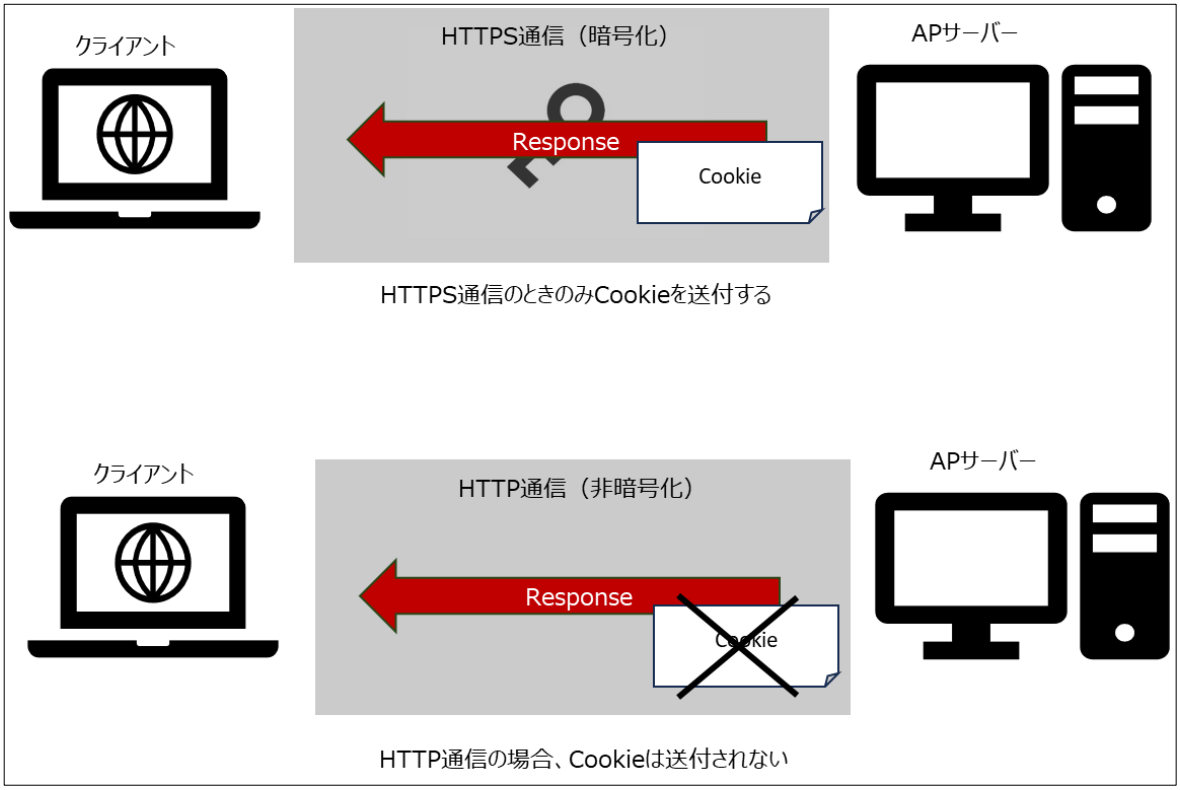

図 13 Cookie のセキュア属性イメージ

OutSystems における Cookie のセキュア属性設定

OutSystems における Cookie のセキュア属性の設定は LifeTime で行うことができます。Cookie のセキュア属性は環境単位でのみ設定が可能です。また CSP と同様 LifeTime を導入していない場合は Service Center で行います。

設定は LifeTime の「環境のセキュリティ」画面から簡単に行うことができます（CSP 設定画面と同じ画面になります）。画面内の Cookies 項目に、Secure を ON/OFF するフラグがありますので、Cookie のセキュア属性を有効にする場合は、トグルスイッチをクリックし右スライドさせます（図 14 Cookie のセキュア属性を ON としたとき の状態になります）。

図 14 Cookie のセキュア属性を ON としたとき

Web ページにおけるセキュア属性の確認

Cookie のセキュア属性を設定するとどうなるかを確認します。

まず、セキュア属性が OFF の場合を確認してみます。CSP と同様、Cookie のセキュア属性は Chrome のデベロッパーツールで確認できます。図 15 の囲った部分が Cookie のセキュア属性の状態を示しています。

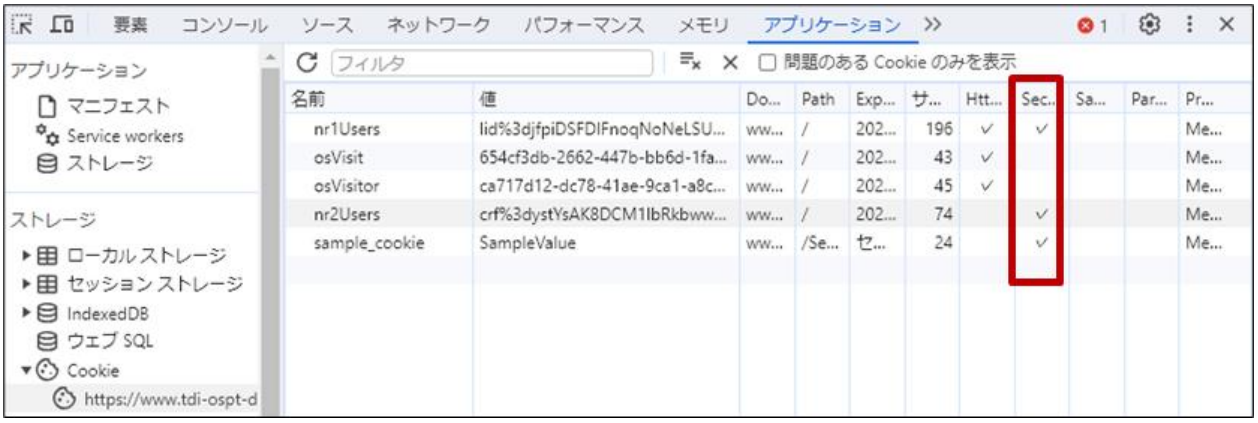

図 15 セキュア属性 OFF の場合

セキュア属性が OFF の場合、Web ページにセキュアでない Cookie が存在することがわかります。

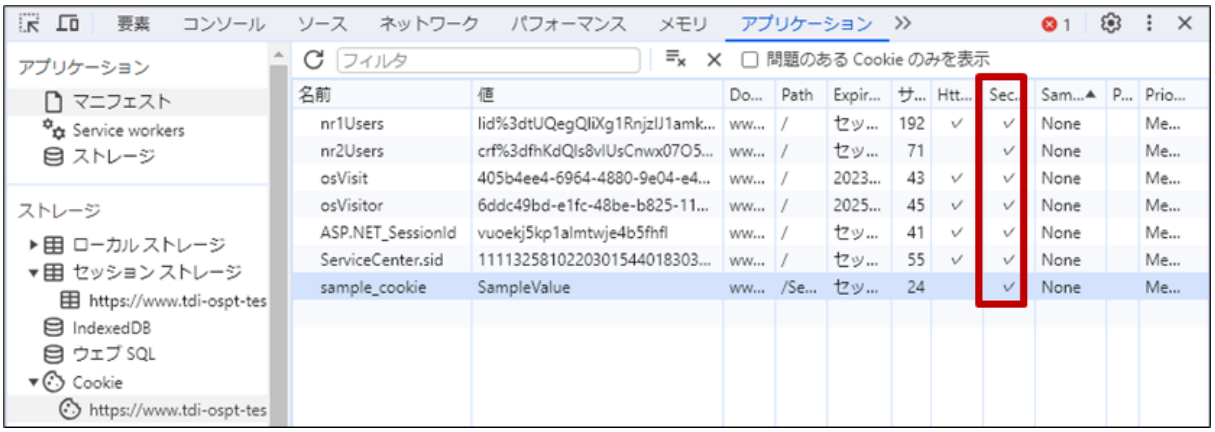

図 16 セキュア属性 ON の場合

セキュア属性が ON の場合、OFF の時とは違いすべてセキュアな Cookie になります。

終わりに

OutSystems では本書で紹介した機能以外にもセキュアなアプリケーションを構築するための対応がとられており、ご紹介した内容はほんの一部です。

tdi では、 OutSystems の機能や技術について十分な知識を持った多くの技術者を有しており、資格保有者数は国内トップクラスです。ローコード開発が一般的に注目される以前（2016 年）から重ねた OutSystems 開発の実績をもとに、IT 戦略コンサルティングや OutSystems 導入から運用までをトータルサポートします。また、お客様に合わせた人財育成や内製化もご支援いたします。本件を含め、何かお困りごとがございましたらどうぞお気軽にお問合せください。

【ローコード開発基盤「OutSystems」】

https://www.tdi.co.jp/outsystems/

【お問い合わせ】

https://tdi.smktg.jp/public/application/add/1095

情報技術開発株式会社
営業本部
東京:〒163-1332 東京都新宿区西新宿六丁目 5 番 1 号　新宿アイランドタワー32 階
TEL : 03-5325-4811 (代表)　FAX 03-5325-4812
中部:〒451-6027 愛知県名古屋市西区牛島町 6 番 1 号 名古屋ルーセントタワー27 階
TEL 052-571-6871（代表）　FAX 052-571-3856
関西:〒530-0005 大阪府大阪市北区中之島二丁目 2 番 7 号 中之島セントラルタワー20 階
TEL.06-6201-7739(代表) FAX.06-6201-7740
九州:〒812-0013 福岡県福岡市博多区博多駅東二丁目 10 番 1 号 福岡ビル S 館 7 階
TEL.092-451-8218(代表) FAX.092-474-7379