EC-CUBE セキュリティガイドライン プラグイン

Version 1.0

2014 年 10 月 15 日

目次

改版履歴

Version 0.1
2014 年 8 月 10 日　　株式会社スピリット・オブ　川口
　　新規作成

Version 0.2
2014 年 8 月 20 日　　株式会社スピリット・オブ　川口
　　校正　誤字・誤記の修正

Version 0.3
2014 年 9 月 1 日　　株式会社スピリット・オブ　川口
　　表現の一部修正、注意点の追加、Appendix の追加

Version 1.0
2014 年 10 月 15 日　　株式会社ロックオン　銭谷
　　外部向けに提供するデータの明確化と利用の制限の表現の一部修正

1.概要

1-1. 本セキュリティガイドラインとは

本書は、EC-CUBE プラグインの開発ならびに利用におけるセキュリティ維持の為に定められた、プラグイン開発者向けのガイドラインです。
このセキュリティガイドラインに従って開発を行う事で、担当者の技術レベル等に依存することなく、画一された安全な枠組みの中で一定水準のセキュリティを維持することを目的としています。

1-2. 主な対象

本書の対象は、主に EC-CUBE のプラグインの設計、開発ならびにカスタマイズを行う開発者としています。
また、サブ対象者として、EC-CUBE のプラグインの企画担当者も確認することを推奨とします。

1-3. 本書の範囲制限

本書は、EC-CUBE プラグインに対してのみのセキュリティガイドラインであり、EC-CUBE モジュール機能や EC-CUBE 本体に対する直接的な拡張機能などは範囲としない。
本書の範囲以外の部分については、それぞれのセキュリティガイドラインに準ずるものとする。

2.セキュリティ確保のために備えるべき機能と注意点

2-1. プラグインプログラム全体

プログラムのコーディング上の注意

EC-CUBE のコーディング規則に準じるものとし、セキュリティ上の脅威が無い事を利用者や第三者が確認しやすいプラグインプログラムとすること。

悪い例１）改行コードが LF に統一されていない。
悪い例２）インデントにタブコードを使用している。

良い例１）EC-CUBE コーディング規約に準じる
良い例２）phpDocumentor に従ったコメントが付与されている。

プラグインのコーディング上の注意

EC-CUBE のプラグインコーディング規則に準じるものとし、セキュリティ上の脅威が無い事を利用者や第三者が確認しやすいプラグインプログラムとすること。特に命名規則やファイル構成、展開方法については規約に準拠が必須となる。

悪い例１）改行コードが LF に統一されていない。
悪い例２）インデントにタブコードを使用している。

良い例１）EC-CUBE コーディング規約に準じる
良い例２）phpDocumentor に従ったコメントが付与されている。

マニュアルの提供

プラグインプログラムの利用に関するマニュアルをモジュールと同様条件の下、提供し、間違った利用や設置によるセキュリティ上の脅威が発生しないように善処する。

悪い例）マニュアルが提供されていない。更新されていない。

良い例）マニュアルが定期的に更新されている。

ファイルの構成

ファイルリストを用意されることが推奨される。特にプラグインの機能によりファイルを複製して、公開ディレクトリ側に呼出プログラムファイル等を設置する場合は、そのプログラムファイルを明示することが強く求められる。
攻撃者による攻撃ファイルの設置やプログラムへの改竄を未然に予防するため、特段の必要性が無い限り、EC-CUBE のディレクトリ構造に準拠する位置にファイルを構成させるものとし、むやみに公開側ディレクトリにファイルを設置せず、必要最低限のファイル複製とする。

また、それらの設置内容をファイルリストで明らかにすることで関連するファイルの状態などを容易に確認出来る状態を作るものとする。
また、プラグインファイル自体は EC-CUBE のプラグインパッケージ仕様により展開されることを前提とした構成とする。
利用者が任意に手動で設置する事は基本的に避けるものとする。

悪い例１）利用者にファイルの複製操作や設置操作を別途求める。
（間違った操作でセキュリティホールを産む可能性がある。）
悪い例２）プラグインのインストール時に、公開側に多数のプログラムファイルを設置しているが、その目的が明らかでは無い。

良い例１）ファイルリストとファイル概要説明がマニュアルで提供されているため、不正なファイルが設置されていない事が確認できる。
良い例２）ファイル設置を行わず、EC-CUBE 標準プログラムに対してフックポイント用い、モード値によって追加機能を付与している。

公開されるポイントの明確化

外部とデータを交換する為に公開側ディレクトリ設置するプログラムが存在する場合、その呼出用プログラムファイルの明示を行う。
また、そのプログラムファイルが通信相手として想定している相手方を明示する。
また公開されるポイントに対して、それぞれに対して「公開／管理者／その他」と分類・明示する事が推奨される。また、それぞれに対してアクセス上の規制をどのように施しているかを明示する事が推奨される。
特に「その他」とは、通信する場合において、その相手が決まっている場合を想定し、何らかの規制をかけている（またはかけるべき）部分を明らかとして攻撃者による想定外の攻撃を未然に防ぐ手立てを行いやすい状態とする。

悪い例）ファイルの設置場所が不明で WAF 等で適切な権限設定ができない。

良い例）マニュアルでファイルの設置場所と通信相手が明示されているため、ファイル改竄検知と.htaccess による通信相手の制限を行える。

2-2. プラグインが扱うデータについて

パラメーターの安全性確保

プログラムファイルが POST または GET によりデータ交換のために入力を受け付けるデータ、並びに受け取るデータについては、下記の対策を施すものとする。

入力フォームの正規化とエラーチェック

EC-CUBE のフォームパラムクラス（SC_FormParam）を使用し、適切な入力フォームデータの正規化とチェック処理を行う。また、エラーチェック処理を必ず呼び出すようにする。

フォームパラメーターの拡張による処理

追加のパラメーター値を追加したい場合には SC_FormParam へのパラメーター追加機能を可能な限り用いることで、入力バリデーターなどを考慮せずとも入力チェックが EC-CUBE 本体プログラム処理内で実行されるようにする。

グローバル変数の直接参照の制限

プログラム内ではグローバル変数である$_POST 変数や$_GET 変数、$_REQUEST 変数、$_COOKIE などを直接参照せず、必ず前述のフォームパラムクラスを通してデータを扱うものとする。
ただし$_SERVER 変数、$_SESSION 変数は、この限りではない。
$_GLOBAL 変数は可能な限り使わない事が推奨される。

利用出来る変数の制限

$_COOKIE 変数および Cookie データ、UserAgent などに依存した確認は信頼性が無いため用いない。また Cookie は技術的には secure オプションにより、ある程度の安全性が担保できるが第三者による検証を困難にしやすい為、利用はさける。

表示をする場合には Smarty のテンプレート処理を通して表示を行う

表示に用いる部分にはデータそのものや変数そのまま、または HTML コードを直接プログラムから出力、または変数に HTML コードそのままを入れて出力してはならない。必ず Smarty のテンプレート処理を通す事により、後述のエスケープ処理のみでチェックと処理が可能になっていることが求められる。

入力を受け付けたデータは必ずエスケープ処理をして表示をする

XSS 脆弱性を生じないよう、変数から表示するデータは必ず表示時には Smarty のエスケープ処理を行うものとする。

外部向けに提供するデータの明確化と利用の制限

EC-CUBE 以外の部分と通信するためにデータ等を出力または加工して提供やデータ交換する場合には、通信先を明らかにした上で交換するデータを明示するものとする。
また EC-CUBE 内のデータを EC-CUBE 以外に渡すまたは提供する場合には、予めプラグインの説明や設定画面に EC-CUBE 以外に渡すデータの内容を明示し、利用者の事前の確認と了承を要するものとする。
特に個人情報などを含む場合は必ずプラグインの設定画面等で確認と了承を管理者に事前に行った上で有効化されるものとし、事前の確認をしていないデータを EC-CUBE 以外に対して提供やデータ交換しないものとする。また、管理者が顧客に対して EC-CUBE 以外に対して提供されるデータの内容について周知を行うようマニュアル等に明記する。
また、プラグインの目的以外に不要なデータをプラグイン内で扱わない事で潜在的な間違い等を防ぐ。

悪い例１）外部からの要求に対して、事前の確認を取っていない EC-CUBE 内のデータを自動データ交換連携するプログラム。
【事前了承無きデータ交換】
悪い例２）「受注データ」として確認を取ってデータ交換しているが、プラグインの目的には本来不要な購入者の個人情報が含まれている。
【不要なデータの取り扱い】
悪い例３）非公式に公式サイト以外で配布されていたプラグインを利用したところ、攻撃者のサーバーに自動的に個人情報が送信されていた。
【ウィルス（ワーム）的なプラグインによる個人情報漏洩】

良い例１）事前に確認と了承を得て、受注データのうち必要な合計金額のみをデータ送信するようにしている。
良い例２）購入者の個人情報のうちプラグインに必要なメールアドレスだけを確認画面を経た上で送信している。

ログの記録の徹底

EC-CUBE 以外の部分と通信する場合には、必ず通信先を明らかにした上でのログを残す。EC-CUBE のログ用関数を利用して記録することで、日時とリクエスト元（ブラウザ等での参照元）プログラムと、そのリクエスト者 IP が正しく記録され、攻撃者が現れた場合の問題解決の助けとする。

悪い例）データ交換するプラグインであるがログ記録が一切無い。

良い例）ログファイルのファイル名に日付を付与して保存している。
plg_example_20140101.log

個人情報の保持の制限

プラグインが扱うデータとして個人情報に属するものを、プラグインにおいて独自に EC-CUBE 内または EC-CUBE 配下のデータベース内ならびに他のサーバーやファイル等に保存することを可能な限り避ける。
扱う場合には、符号化（ハッシュ）や、番号化（ＩＤ）、またはステータスなどの間接情報のみを扱い、直接保存しないことが求められる。またログ出力に関しても同様とする。

悪い例１）プラグインが会員データの複製データを別テーブルに保存していたが、会員の退会や削除と連動していなかった。顧客からの求めにより名簿から削除したつもりが、残っていてプラグインの機能によりメルマガが削除後も送信されていた。
【個人情報保護法のオプトアウトに関する事項に違反】

悪い例２）ログに取り扱うデータの記録を全て出力していたため、顧客の個人情報がログファイルに出力されていた。使用していたレンタルサーバーの仕様で読み書き可能なテキストファイルはサーバー内の他のユーザーでも閲覧出来る状態であったため、顧客情報が漏洩してしまった。
【個人情報漏洩】

良い例１）通常時は最低限の出力となっており、定義値でデバッグモード設定をした場合のみログファイルに詳細出力がされるようになっており、開発時は動作検証がスムーズに出来た上に、設定を戻すだけで適切な表示に戻すことが出来たため、他のプラグインとの衝突問題解決がスムーズに出来た。

良い例２）会員情報を拡張するため別テーブルをプラグインで用意して保存をしているが、会員 ID と符号化した情報だけを保存している。

生成ファイルの設置・保存場所の隠蔽

プラグインが生成または設置または一時的にでも取り扱うデータのファイルについては、公開ディレクトリ側に可能な限り設置しないことが求められる。特に、CSV ファイルの生成やアップロード時の保存先指定として、公開ディレクトリ側に設置した場合、悪意のある第三者からも閲覧可能になる可能性があるため、本当に公開して良い画像ファイル類のみを設置することが求められる。

悪い例１）ファイルを処理後にすぐに削除するからと、アップロードする CSV の一時保存先に公開ディレクトリを指定している。処理途中に意図せぬエラーが生じて処理中断された場合にファイル削除が働かず、アップロードした CSV ファイルが公開ディレクトリ側に残っていた。アップロード対処の CSV ファイルは個人情報が含まれていて、後日悪意のある第三者に閲覧されてしまった。【個人情報漏洩】

悪い例２）プラグインが購入情報を加工して CSV ファイルを生成してダウンロードできる機能を提供していたが、生成するファイルの置き場所を公開ディレクトリ側に指定していて、削除されていなかった CSV ファイルが残っていた。残っていた CSV ファイルを後日悪意のある第三者に閲覧されてしまった。【個人情報漏洩】

良い例）プラグインが扱うファイルは data/downloads/ディレクトリ data/upload/ディレクトリの下に全て設置しており安全な領域になっている。また、削除処理などを適切に行っている。

2-3. プラグインが行うプログラム介入について

テンプレート利用の徹底

画面表示に関わる部分に対して追加の表示を行う場合は、以下の介入処理または追加処理を必ず利用するものとし、プラグインが独自に行う事は可能な限り避ける。

・テンプレートをフック（prefilterTransform または outputfilter_taransform）
・ヘッダーにタグを追加（setHeadNavi）
・ブロックを追加
・ページを追加

テンプレート処理を用いず表示する事は可能な限り避け、テンプレート処理を用いることが出来ない部分については、同等のエスケープ処理などを用いる。

悪い例）介入処理を用いずに表示をしたため、エスケープ処理が足りず不正なコードを表示出来てしまった。
【XSS 脆弱性や CSRF 脆弱性】

良い例）適切にテンプレート内にフックと編集が出来るようになっていたため、エスケープ漏れが無いかの確認が速やかに出来た。

※Appendix　1.テンプレート利用の徹底例

変数名の固有化

プラグインが用いる独自のテンプレート変数など他でも参照・代入が出来る変数は、プラグインの名前を付与して固有の変数名とし、他のプラグイン等と被らないようにする。

悪い例）テンプレート変数として$data_id という変数に価格を代入するようにしたが、他のプラグインも同名を利用していたため、想定していた値が書き換わってしまい、本来見えてはいけない他の顧客データが表示されてしまった。

良い例）テンプレート変数は$plg_example_data_id としているため、他と混同して利用する事が無い。また、データ読み出し時にも会員 ID との紐付けをチェックしている。

不要な加入をしない

必要以上の介入処理を行わず、必要なタイミングのみ介入とデータ操作を行う事が求められる。これにより、想定されていないデータの投入による誤作動などを防がれる。

悪い例）テンプレートをフック（prefilterTransform）していて、入力されたデータに応じてデータ置換して表示をするようにしていたが、介入すべきテンプレート以外でもフックしていたため、同名の変数を使っていた所に、他の顧客のデータを出力してしまった。
【個人情報漏洩】

良い例）追加の表示用情報を表示するべきタイミングのみで、トランスフォーム機能を呼び出して HTML ソースを改変している。

※Appendix　2.不要な介入の例を参照

3.各種セキュリティ対策ガイド

以下にプラグインで特に気をつけるべきセキュリティ対処事例を挙げ、プラグインにおいては、これらの対策を適切に施すものとする。

外部からの通知の制限

悪い例）パラメーターの書式があっていれば受け付けてしまう。
【セッションハイジャックの一種が生じる可能性】

良い例）外部からの通知などを受け付ける際に、接続元 IP アドレスを制限している。

エラー出力

悪い例）外部から与えられたエラー内容をそのまま表示している。エスケープが十分な場合でも表示される事が不適切な情報が含まれる場合を考慮していない。

良い例）エラー出力は、エラーコードとモジュール内に用意されたコードから表示される文字列の対照データによって出力内容が固定的になっている。

※十分に一般化されたエラー表示内容を信頼出来る相手から得られている場合はこの限りでは無い。

過度なセッションの会員情報信頼

悪い例）会員 ID だけを信頼して取り扱ってしまい、会員 ID が DB の障害やロールバック処理などにより重複する場合などを想定していない。
同じ会員 ID が与えられた会員が万が一、居た場合に他の会員情報を用いて処理を実行してしまう。

良い例）会員 ID とセットで会員データ毎にランダムに与えられる secret_key の一致をチェックしている。

Appendix

1. **テンプレート利用の徹底例**

以下、プラグインが何らかの処理により出力をする場面を想定している。

悪いコードの例

```
1    $output_data = ‘テストに移動します’ . $_REQUEST[‘data’];
2    $output_data .= ‘<script>setTimeout(function(){location.href=” . $url;
3    $output_data .= ‘”;});</script>’;
4    echo $output_data;
5    exit;
```

解説

テンプレートを利用せず、単なる移動スクリプトだからといって直接 echo して画面に出力をしている。エスケープ処理なども不適切である。

良いコードの例

・テンプレートファイル

```
1    <p>テストに移動します。<!—{$plg_example_output|h}→</p>
2    <script> function plgExampleMove() {
3                                    var url = ‘<!—{$plg_example_url|h}→’;
4                                    location.href = url;
5            }
6    </script>
```

・対応する PHP 処理の抜粋

```
1    $objPage->plg_example_output = $objFormParam->getValue(‘data’);
2    $objPage-> plg_example_url| = $url;
3    $objPage->tpl_onload .= ‘plgExampleMove();’;
```

解説

テンプレートファイルと PHP ファイルに処理が分離されている。
テンプレート側は出力内容と対応する JavaScript の関数のみ用意されていてテンプレート変数はエスケープされて表示されることが明示されている。
PHP 処理側は、変数をテンプレート変数に設定し、さらに標準の自動読み込みの変数に追記する形で実行しており、大変スマートかつ安全な状態である。

2. 不要な介入の例

以下、prefilterTransform フックポイントから prefilterTransform 関数が呼ばれるという設定状態にあることを例にして取り扱う。

悪いコードの極端な例

```
1    function prefilterTransform(&$source, LC_Page_Ex $objPage, $filename) {
2        $objTransform = new SC_Helper_Transform($source);
4        $objTransform->select('select')->appendChild('<option>'
5                                                    . $_REQUEST['sample']);
6        $source = $objTransform->getHTML();
7    }
```

解説

上記のコードでは、全てのテンプレートに必ず介入してしまう。また、パラメーターをそのままテンプレートに差し込もうとしている。プラグイン単体と特定のページだけで見れば動作上の影響は少ないが、実際には他のプラグインや複数の様々なページから呼び出されることを考えると、大変不適切である。

悪いコードの例

```
1    function prefilterTransform(&$source, LC_Page_Ex $objPage, $filename) {
2        $objTransform = new SC_Helper_Transform($source);
3        if (strpos($filename, 'customer/edit.tpl') !== false) {
4            $objTransform->select('table.form')->appendChild('<br/>');
5        }
6        $source = $objTransform->getHTML();
7    }
```

解説

上記のコードでは介入するテンプレートを制限しているため一見問題無いように見える。しかしながら、if 文の外に 6 行目の処理があるため、全てのテンプレート処理に対して必要が無いトランスフォームの処理を通していることとなる。全ての画面での動作を確認しているのであれば良いが実質的に不可能で有り、予期せぬ介入により全体の動作やセキュリティへの障害となる可能性を含んでいる。

良いコードの例

```
1	function prefilterTransform(&$source, LC_Page_Ex $objPage, $filename) {
2		if (strpos($filename, 'customer/edit.tpl') !== false) {
3			$objTransform = new SC_Helper_Transform($source);
4			$objTransform->select('table.form')->appendChild(‘<br/>’);
5			$source = $objTransform->getHTML();
6		}
7	}
```

解説

上記コードの例では if 文で介入の範囲が完全に絞られており、処理も限定されている。このような処理では動作確認する範囲も絞られるため、問題が生じにくい。またカスタマイズする人にとっても意図せぬ問題が生じにくい