Fortify Audit Workbench

# Developer Workbook

## DB-Card-main

# Table of Contents

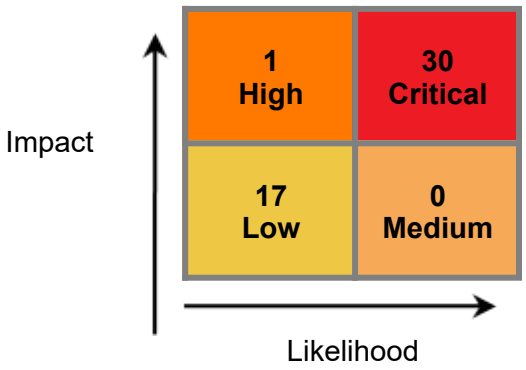
# Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the DB-Card-main project audit. The information contained in this workbook is targeted at project managers and developers.

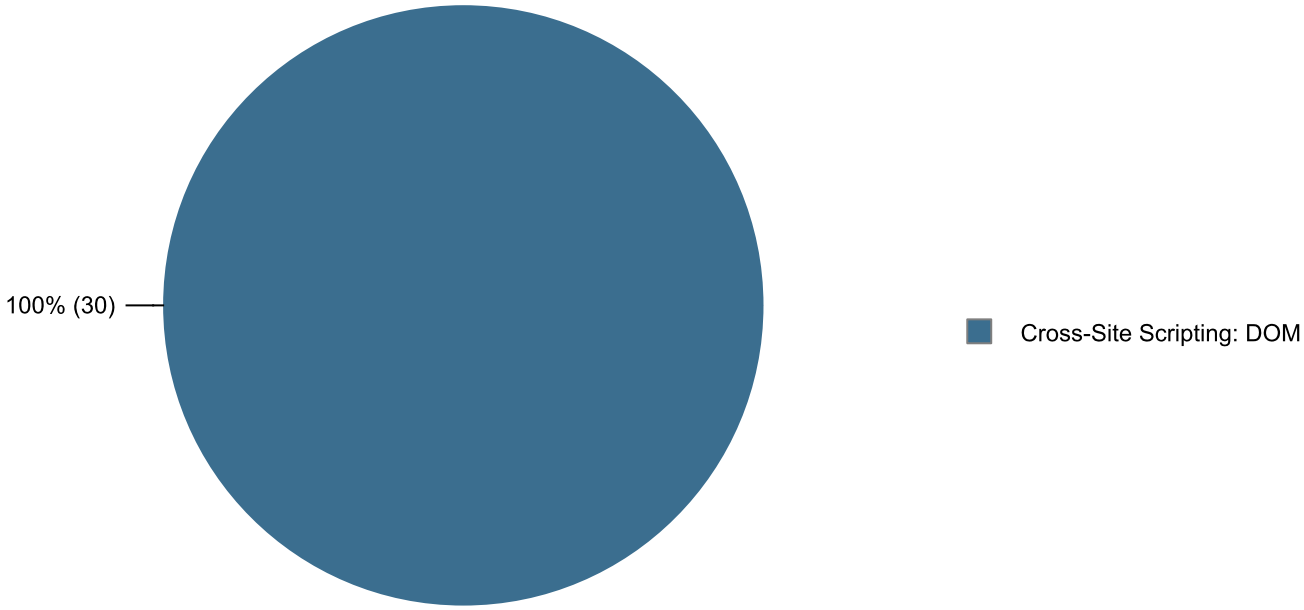
This section provides an overview of the issues uncovered during analysis.

| | |
|---|---|
| **Project Name:** | DB-Card-main |
| **Project Version:** | |
| **SCA:** | Results Present |
| **WebInspect:** | Results Not Present |
| **WebInspect Agent:** | Results Not Present |
| **Other:** | Results Not Present |

**Issues by Priority**




**Top Ten Critical Categories**



# Project Description

This section provides an overview of the Fortify scan engines used for this project, as well as the project meta-information.

## SCA

| | | | |
|---|---|---|---|
| **Date of Last Analysis:** | Sep 5, 2025 11:14 AM | **Engine Version:** | 24.4.1.0005 |
| **Host Name:** | IISIFTFYSRV02 | **Certification:** | VALID |
| **Number of Files:** | 31 | **Lines of Code:** | 13,947 |

| Rulepack Name | Rulepack Version |
|---|---|
| Fortify 安全編碼規則、社群、Cloud | 2025.2.1.0001 |
| Fortify 安全編碼規則、社群、Universal | 2025.2.1.0001 |
| Fortify 安全編碼規則、核心、Cloud | 2025.2.1.0001 |
| Fortify 安全編碼規則、核心、JavaScript | 2025.2.1.0001 |
| Fortify 安全編碼規則、核心、Universal | 2025.2.1.0001 |
| Fortify 安全編碼規則、延伸、配置 | 2025.2.1.0001 |
| Fortify 安全編碼規則、延伸、內容 | 2025.2.1.0001 |
| Fortify 安全編碼規則、延伸、JavaScript | 2025.2.1.0001 |

# Issue Breakdown by Fortify Categories

The following table depicts a summary of all issues grouped vertically by Fortify Category. For each category, the total number of issues is shown by Fortify Priority Order, including information about the number of audited issues.

| Category | Fortify Priority (audited/total) | | | | Total Issues |
|---|---|---|---|---|---|
| | Critical | High | Medium | Low | |
| Cross-Site Request Forgery | 0 | 0 | 0 | 0 / 2 | 0 / 2 |
| Cross-Site Scripting: DOM | 0 / 30 | 0 | 0 | 0 | 0 / 30 |
| Cross-Site Scripting: Poor Validation | 0 | 0 | 0 | 0 / 12 | 0 / 12 |
| Cross-Site Scripting: Self | 0 | 0 | 0 | 0 / 2 | 0 / 2 |
| Open Redirect | 0 | 0 / 1 | 0 | 0 | 0 / 1 |
| System Information Leak: Internal | 0 | 0 | 0 | 0 / 1 | 0 / 1 |

# Results Outline

## Cross-Site Request Forgery (2 issues)

### Abstract

表單發佈必須包含特定使用者機密，以避免攻擊者作出未經授權的要求。

### Explanation

防禦跨網站偽造要求 (CSRF) 弱點會在以下情況中出現： 1. Web 應用程式使用了階段作業 Cookie。 2. 應用程式在回應 HTTP 要求時，沒有驗證該要求是否經使用者同意產生。 nonce 是與訊息一起傳送的加密隨機值，用於防止重複進行的攻擊。如果 HTTP 要求目前沒有包含其來源的證明，則負責處理 HTTP 要求的程式碼會容易受到 CSRF 的攻擊 (除非它不改變應用程式的狀態)。這代表使用階段作業 Cookie 的 Web 應用程式必須特別留意，以確保攻擊者沒辦法傳遞假造的要求去欺騙使用者。想像若 Web 應用程式允許管理人員提交以下表單建立新帳戶：

```
<form method="POST" action="/new_user" >
Name of new user: <input type="text" name="username">
Password for new user: <input type="password" name="user_passwd">
<input type="submit" name="action" value="Create User">
</form>
```

攻擊者可能建立如以下的網站：

```
<form method="POST" action="http://www.example.com/new_user">
<input type="hidden" name="username" value="hacker">
<input type="hidden" name="user_passwd" value="hacked">
</form>
<script>
document.usr_form.submit();
</script>
```

如果 example.com 管理員在網站上的有效階段作業期間造訪惡意頁面，則會在不知不覺中為攻擊者建立帳戶。這就是 CSRF 攻擊。這樣是有可能的，因為應用程式無法確定要求的來源。因此不論是使用者建立的或是由攻擊者建立的偽造操作，都會被視為合法的操作。攻擊者不會看到假造的要求所產生的網頁，所以這種攻擊技術只有對修改應用程式狀態的要求有用。 在 URL 中傳送階段作業識別碼而非當作 Cookie 的應用程式不會有 CSRF 問題，因為攻擊者沒有辦法存取階段作業識別碼做為假造的要求。

### Recommendation

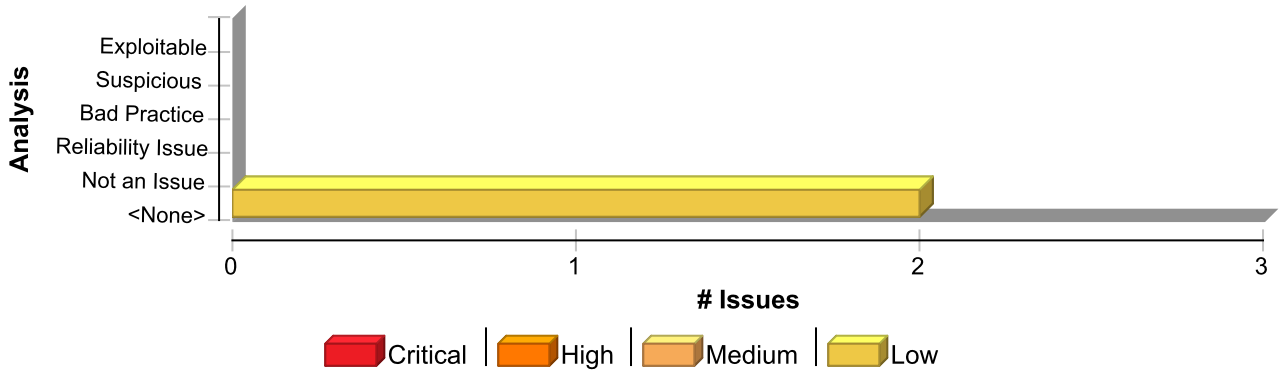
使用階段作業 Cookie 的應用程式必須包含每個發佈表單的一部分資訊，以供後端程式碼用來驗證要求的來源。其中一種方法是包含一個隨機要求識別碼或 nonce，如下所示：

```
RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
body = addToPost(body, new_username);
body = addToPost(body, new_passwd);
body = addToPost(body, request_id);
rb.sendRequest(body, new NewAccountCallback(callback));
```

之後後端邏輯就能夠在處理其他的表單資料前，先驗證要求識別碼。如果可能，各伺服器要求應具備唯一的要求識別碼，而非讓特定工作階段的所有要求共用要求識別碼。就階段作業識別碼而言，攻擊者如果愈難猜出要求識別碼，就愈難發起一次成功的 CSRF 攻擊。權杖應無法輕易猜出，且其受保護的方式應與保護工作階段權杖的方式相同 (例如使用 SSLv3)。 其他減緩技術包括： **架構保護：**大多數現代 Web 應用程式架構都內嵌 CSRF 保護，且會自動包括並驗證 CSRF 權杖。 **使用挑戰/回應控制項：**強制客戶回應伺服器傳送的挑戰是針對 CSRF 的強大防禦措施。可用於此用途的一些挑戰有：CAPTCHA、密碼重新驗證和一次性權杖。 **檢查 HTTP Referer/Origin 表頭：**攻擊者將無法在執行 CSRF 攻擊時欺騙這些表頭。因此，這些表頭將成為防止 CSRF 攻擊的有用方式。 **對階段作業 Cookie 進行雙重提交：**除了傳送實際的階段作業 ID Cookie 外，還以隱藏表單值的形式傳送階段作業 ID Cookie，這是針對 CSRF 攻擊的良好保護方式。伺服器會先檢查這兩個值，確保其完全相同，然後再處理其餘的表單資料。若攻擊者代表使用者提交表單，將無法根據相同來源策略修改階段作業 ID Cookie 值。 **限制階段作業存留期：**若使用 CSRF 攻擊存取受保護的資源，只有在作

為攻擊的一部分而傳送的階段作業 ID 在伺服器上仍有效時，攻擊才有效。限制階段作業存留期會降低攻擊成功的可能。 XSS 攻擊會破解此處描述的技術。有效的 CSRF 減緩措施包括 XSS 減緩技術。

## Issue Summary




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| Cross-Site Request Forgery | 2 | 0 | 0 | 2 |
| **Total** | **2** | **0** | **0** | **2** |

## Cross-Site Request Forgery — Low

### Package: <none>

### nfc-generator-bilingual.html, line 273 (Cross-Site Request Forgery)

#### Issue Details

**Kingdom:** Encapsulation
**Scan Engine:** SCA (Content)

#### Sink Details

**Sink:**
**File:** nfc-generator-bilingual.html:273
**Taint Flags:**

```
270  </div>
271
272  <div class="form-container">
273  <form id="cardForm">
274  <div class="form-group">
275  <label>版面類型：</label>
276  <select id="layoutType">
```

### nfc-generator.html, line 191 (Cross-Site Request Forgery)

#### Issue Details

**Kingdom:** Encapsulation
**Scan Engine:** SCA (Content)

#### Sink Details

## Cross-Site Request Forgery — Low

### Package: <none>

### nfc-generator.html, line 191 (Cross-Site Request Forgery)

**Sink:**
**File:** nfc-generator.html:191
**Taint Flags:**

```
188  </div>
189
190  <div class="main-content">
191  <form id="cardForm">
192  <div class="form-section">
193  <h2>版面設定</h2>
194  <div class="form-group">
```

## Cross-Site Scripting: DOM (30 issues)

### Abstract

傳送未經驗證的資料至網路瀏覽器，會導致瀏覽器執行惡意的程式碼。

### Explanation

Cross-site scripting (XSS) 弱點會在以下情況中出現： 1.資料從一個不可信賴的來源進入 Web 應用程式。在基於 DOM 的 XSS 案例中，會從 URL 參數或瀏覽器內的其他值來讀取資料，並以用戶端程式碼回寫到頁面中。在 Reflected XSS 案例中，不可信賴的來源通常為網頁要求，而在 Persisted XSS (也可稱為 Stored XSS) 案例中，來源通常為資料庫或其他後端資料儲存區。 2.未經驗證且包含在動態內容中的資料將傳送給某個網頁使用者。在基於 DOM 的 XSS 案例中，一旦受害者的瀏覽器開始剖析 HTML 頁面，就會開始執行惡意內容，且成為建立 DOM (Document Object Model，文件物件模型) 的一部分。 傳送到網頁瀏覽器的惡意內容經常是以 JavaScript 片段的形式出現，但是也可能包含 HTML、Flash 或者瀏覽器執行的任何其他程式碼類型。以 XSS 為基礎的攻擊手段花樣百出且幾乎無窮無盡，但是它們通常會傳輸 Cookie 或其他階段作業資訊之類的私人資料給攻擊者、將受害者重新導向到攻擊者控制的網頁內容，或者利用易受攻擊的網站，在使用者的機器上執行其他惡意操作。 **範例 1：**以下的 JavaScript 程式碼片段會從 URL 讀取員工識別碼 `eid` 並顯示給使用者。

```
<SCRIPT>
var pos=document.URL.indexOf("eid=")+4;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
```

**範例 2：**請考慮使用 HTML 表單：

```
<div id="myDiv">
Employee ID: <input type="text" id="eid"><br>
...
<button>Show results</button>
</div>
<div id="resultsDiv">
...
</div>
```

以下的 jQuery 程式碼片段會從表單讀取員工識別碼，並顯示給使用者。

```
$(document).ready(function(){
$("#myDiv").on("click", "button", function(){
var eid = $("#eid").val();
$("resultsDiv").append(eid);
...
});
});
```

如果文字輸入中的員工識別碼 (識別碼為 `eid`) 只包含標準英數字元，這些程式碼便會正確地運作。如果 `eid` 中有包含中繼字元或來源程式碼中的值，那麼網路瀏覽器就會像顯示 HTTP 回應那樣執行程式碼。 **範例 3：**以下程式碼顯示了 React 應用程式內基於 DOM 的 XSS 範例：

```
let element = JSON.parse(getUntrustedInput());
ReactDOM.render(<App>
{element}
</App>);
```

在 `Example 3` 中，如果攻擊者可以控制從 `getUntrustedInput()` 擷取的整個 JSON 物件，則可能會讓 React 將 `element` 解譯為元件，因此可以傳遞具有 `dangerouslySetInnerHTML` 及其自己的控制值的物件 (典型的 Cross-Site Scripting 攻擊)。 一開始，這些似乎不會輕易受到攻擊。畢竟，誰會提供包含在自己電腦上執行的惡意程式碼的輸入？其實，真正的危險在於攻擊者會建立惡意的 URL，接著使用電子郵件或社交工程病毒誘騙受害者透過連結前往該 URL。當受害者按下該連結時，他們即不知不覺地透過易受攻擊的 Web 應用程式，將惡意內容資訊帶回他們自己的電腦。這個利用易受攻擊的 Web 應用程式進行攻擊的機制就是通常所知的 Reflected XSS。 如同範例中所示，XSS 的弱點是由 HTTP 回應中包含未經驗證資料的程式碼所引起的。XSS 攻擊有三種途徑可攻擊受害者： - 直接從 HTTP 要求讀取資料，並直接回傳至 HTTP 回

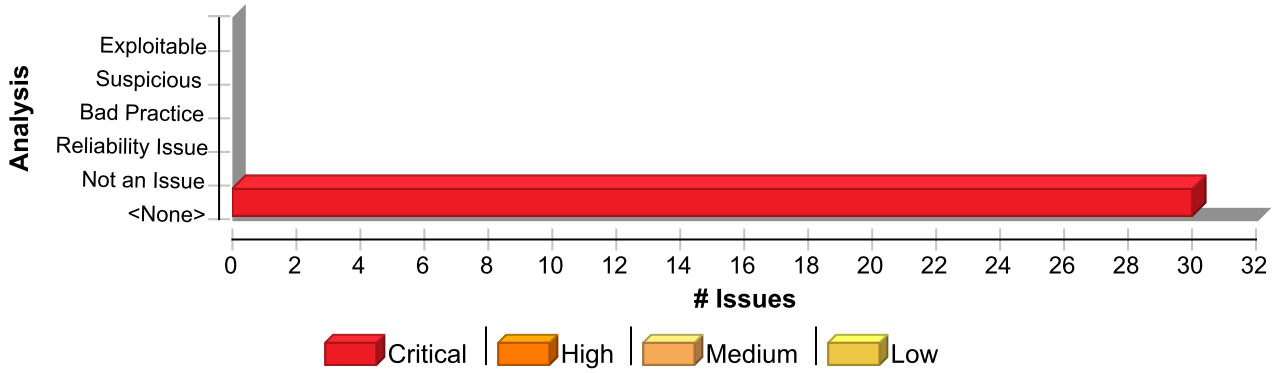
應。當攻擊者誘使使用者提供危險內容給易受攻擊的 Web 應用程式，接著這些危險內容就會回傳給使用者並由在網路瀏覽器中執行，這時就會出現 Reflected XSS 攻擊行為。傳遞惡意內容最常用的機制就是，將惡意內容當作參數隱藏在公開發表的 URL 中，或者以電子郵件方式直接傳送給受害者。以這種方法建立的 URL 會構成許多網路釣魚 (phishing) 架構的核心，攻擊者可以藉此誘使受害者去造訪一個指向到易受攻擊網站的 URL。網站將攻擊者的內容回傳給使用者之後，就會執行這些內容，並接著從使用者的電腦將可能包含階段作業資訊的 Cookie 之類的私人資訊傳給攻擊者，或者執行其他惡意活動。 - 應用程式會將危險資料儲存到資料庫或其他可信任的資料儲存中。這些危險資料隨後會被回讀到應用程式，並包含在動態內容中。Persistent XSS 攻擊會在以下情況出現：攻擊者把危險內容插入到之後會讀取的資料記憶體中，並包含在動態內容中。從攻擊者的角度來看，插入惡意內容的最佳位置莫過於一個會對很多使用者或特別感興趣的使用者顯示的區域。感興趣的使用者通常會在應用程式中擁有較高的權限，或者會與敏感資料進行互動，且這些資料對攻擊者而言很有利用價值。如果其中一個使用者執行了惡意內容，攻擊者可能會代替使用者去執行需要權限許可的作業，或者取得存取使用者專屬敏感資料的權限。 - 應用程式以外的來源會在資料庫或是其他資料記憶體中儲存危險資料，且之後這些危險資料會被當作信賴的資料回讀到應用程式，並會包含在動態內容中。

## Recommendation

防止 XSS 的解決方法是：確保在需要的位置進行驗證，並設定相關屬性以防止漏洞。 由於 XSS 弱點會在應用程式於輸出中包含惡意資料時出現，所以合理方法就是，在應用程式輸出資料之前馬上驗證資料。然而，由於 Web 應用程式經常會為了產生動態內容而包含複雜且難以理解的程式碼，所以此方法容易遺漏錯誤 (遺漏驗證)。降低此風險的有效方法即為執行 XSS 輸入驗證。 Web 應用程式必須驗證所有輸入以防止出現其他弱點 (如 SQL Injection)，因此比較簡單的方法就是，加強應用程式現有的輸入驗證機制，納入 XSS 檢查。儘管有一定的作用，但是 XSS 的輸入驗證並不能取代嚴格的輸出驗證。應用程式可能會透過共用資料存放區或者其他信賴的來源接受輸入，而該資料存放區可能會從未執行適當輸入驗證的來源接受輸入。因此，應用程式不能間接依賴此資料或任何其他資料的安全性。這代表杜絕 XSS 弱點的最佳方法就是，驗證所有進入應用程式的資料，以及所有離開應用程式並傳送到使用者的資料。 驗證 XSS 最安全的方法，是建立一個安全字元的允許清單，只允許清單上的字元可以出現在 HTTP 內容，並僅接受由經過檢驗的集合中字元組成的輸入。例如，有效的使用者名稱可能僅包含英數字元，或電話號碼可能僅包含數字 0-9。然而，這種解決方式在 Web 應用程式中經常是不可行的，因為很多字元對瀏覽器來說都具有特殊意義，將這些字元編碼之後，必須將它們視為有效輸入。例如，一個網站設計公告欄就必須接受來自於使用者的 HTML 片段。 更具彈性但安全性較低的方法是實作拒絕清單，能在使用輸入前選擇性地拒絕或避開可能有危險的字元。若要建立這類名單，首先必須了解那些對網頁瀏覽器來說有特殊意義的字元集。雖然 HTML 標準會定義具有特殊意義的字元，但是許多網頁瀏覽器會嘗試修正 HTML 中的常見錯誤，而且在特定環境中可能會將其他字元視為有特殊意義。這就是為什麼我們不建議使用拒絕清單做為預防 XSS 的方式。Carnegie Mellon 大學軟體工程學院 (Software Engineering Institute) 下的 CERT(R) 合作中心 (CERT(R) Coordination Center) 提供了在各種環境中具有特殊意義的字元資訊 [1]： 在區塊等級 (block-level) 元素的內容中 (位於文字段落中間)： -「<」是特殊字元，因為它會引入標籤。 -「&」是特殊字元，因為它會引入字元實體。 -「>」是特殊字元，因為某些瀏覽器將其視為特殊字元，會假設該頁面的作者想加入開頭的「<」，卻不小心遺漏掉了。 以下原則適用於屬性值： - 在以雙引號括住的屬性值中，雙引號之所以特殊，是因為它們標記了屬性值的結尾。 - 在以單引號結尾的屬性值中，單引號是特殊字元，因為它們標記了屬性值的結尾。 - 在沒有任何引號的屬性值中，空格字元 (例如空格和定位字元) 也是特殊字元。 -「&」和特定屬性一起使用時會變成特殊字元，因為它會引導出一個字元實體。 舉例來說，在 URL 中，搜索引擎可能會在結果頁面提供一個連結，讓使用者可以按一下該連結來重新執行搜尋。這個方法可以運用在編寫 URL 中的搜尋查詢，此動作會引導出其他特殊字元： - 空格、定位字元和換行符號都是特殊字元，因為它們標記了 URL 的結尾。 -「&」為特殊字元，因為它會引導出一個字元實體或分隔 CGI 參數。 - 非 ASCII 字元 (就是所有在 ISO-8859-1 編碼表中大於 127 的字元) 不允許出現在 URL 中，所以它們在此內容中被視為特殊字元。 - 每當伺服器端程式碼對以 HTTP 逸出序列編碼的參數進行解碼時，就必須從輸入中篩選 "%" 符號。例如，對於諸如「%68%65%6C%6C%6F」的輸入，必須篩選「%」，才能在網頁上顯示「hello」。 在 的正文中： - 將文字直接插入原有指令碼標籤時，必須篩選分號、括號、中括號以及換行字元。 伺服器端 Script： - 如果伺服器端 Script 會將輸入中的任何驚嘆號字元 (!) 轉換為輸出中的雙引號字元時 (")，則可能需要進行更多篩選。 其他可能性： - 若攻擊者提交 UTF-7 編碼的要求，特殊字元「<」會顯示為「+ADw-」，並且可能會避開篩選。如果輸出包含在沒有明確指定編碼格式的頁面中，某些瀏覽器會嘗試以智慧方式來根據內容識別編碼 (在此情況下為 UTF-7)。 在您確定在應用程式中針對 XSS 攻擊執行驗證的正確要點，以及驗證時要考慮的特殊字元後，那麼下一個挑戰就是決定驗證過程中處理特殊字元的方法。如果應用程式將某些特殊字元認定為無效輸入，那麼您可以拒絕任何包含這些被視為無效特殊字元的輸入。第二種選擇是以篩選方式移除特殊字元。然而，篩選所產生的副作用在於會改變篩選內容的顯示樣貌。但是在需要完整顯示輸入內容時，這種情況是無法接受的。 如果必須接受包含特殊字元的輸入且必須準確顯示輸入內容，驗證動作一定要編碼所有特殊字元，以移除特殊字元代表的意義。官方 HTML 規格 [2] 提供了特殊字元對應的 ISO 8859-1 編碼值完整清單。 很多應用程式伺服器都試圖避免應用程式出現

Cross-site scripting 弱點，方法是為負責設定某個特定 HTTP 回應內容的函數提供各種實作，以驗證是否存在進行 Cross-site scripting 攻擊的字元。請勿依賴執行應用程式的伺服器來確保應用程式的安全性。對於任何開發的應用程式，無法保證在其存留期內將在哪些應用程式伺服器上執行。隨著標準和已知盜取方式不斷地演變，無法保證應用程式伺服器會繼續保持同步。

## Issue Summary




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| Cross-Site Scripting: DOM | 30 | 0 | 0 | 30 |
| **Total** | **30** | **0** | **0** | **30** |

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-en.html, line 746 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index-en.html:746

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-en.html, line 746 (Cross-Site Scripting: DOM)

**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
743  }
744  // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
745  const tempDiv = document.createElement('div');
746  tempDiv.innerHTML = processedContent;
747  while (tempDiv.firstChild) {
748  socialInfoContent.appendChild(tempDiv.firstChild);
749  }
```

### index-personal-en.html, line 690 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index-personal-en.html:690
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
687  }
688  // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
689  const tempDiv = document.createElement('div');
690  tempDiv.innerHTML = processedContent;
691  while (tempDiv.firstChild) {
692  socialInfoContent.appendChild(tempDiv.firstChild);
693  }
```

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index-personal.html, line 690 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784   }
785
786   function getCardDataFromNFC() {
787   const urlParams = new URLSearchParams(window.location.search);
788   let cardDataParam = urlParams.get('data');
789   let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index-personal.html:690
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
687   }
688   // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
689   const tempDiv = document.createElement('div');
690   tempDiv.innerHTML = processedContent;
691   while (tempDiv.firstChild) {
692   socialInfoContent.appendChild(tempDiv.firstChild);
693   }
```

### index.html, line 722 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

```
889
890   function getCardDataFromNFC() {
891   // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
```

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index.html, line 722 (Cross-Site Scripting: DOM)

```
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index.html:722
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
719  }
720  // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
721  const tempDiv = document.createElement('div');
722  tempDiv.innerHTML = processedContent;
723  while (tempDiv.firstChild) {
724  socialInfoContent.appendChild(tempDiv.firstChild);
725  }
```

### index1-en.html, line 746 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index1-en.html:746
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index1-en.html, line 746 (Cross-Site Scripting: DOM)

```
743  }
744  // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
745  const tempDiv = document.createElement('div');
746  tempDiv.innerHTML = processedContent;
747  while (tempDiv.firstChild) {
748  socialInfoContent.appendChild(tempDiv.firstChild);
749  }
```

### index1.html, line 722 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to tempDiv.innerHTML
**Enclosing Method:** renderCard()
**File:** index1.html:722
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
719  }
720  // 由於 processSocialLinks 返回 HTML 字符串，我們需要安全地解析它
721  const tempDiv = document.createElement('div');
722  tempDiv.innerHTML = processedContent;
723  while (tempDiv.firstChild) {
724  socialInfoContent.appendChild(tempDiv.firstChild);
725  }
```

### index-en.html, line 691 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index-en.html, line 691 (Cross-Site Scripting: DOM)

**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index-en.html:691
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
688
689  avatar.alt = data.name;
690  console.log('Loading avatar:', data.avatar);
691  avatar.src = data.avatar;
692
693  // 聯絡資訊
694  const emailLink = document.getElementById('user-email');
```

### index-personal-en.html, line 627 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
```

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index-personal-en.html, line 627 (Cross-Site Scripting: DOM)

```
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index-personal-en.html:627
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
624  this.style.display = 'none';
625  };
626  avatar.alt = data.name;
627  avatar.src = data.avatar;
628
629  // Contact information
630  const emailLink = document.getElementById('user-email');
```

### index-personal.html, line 627 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index-personal.html:627
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
624  this.style.display = 'none';
625  };
626  avatar.alt = data.name;
627  avatar.src = data.avatar;
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-personal.html, line 627 (Cross-Site Scripting: DOM)

```
628
629  // 聯絡資訊
630  const emailLink = document.getElementById('user-email');
```

### index.html, line 661 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index.html:661
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
658
659  // 開始載入圖片
660  console.log('Loading avatar:', data.avatar);
661  avatar.src = data.avatar;
662
663  // 聯絡資訊
664  const emailLink = document.getElementById('user-email');
```

### index1-en.html, line 691 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index1-en.html, line 691 (Cross-Site Scripting: DOM)

**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895   }
896
897   function getCardDataFromNFC() {
898   const urlParams = new URLSearchParams(window.location.search);
899   let cardDataParam = urlParams.get('data');
900   let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index1-en.html:691
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
688
689   avatar.alt = data.name;
690   console.log('Loading avatar:', data.avatar);
691   avatar.src = data.avatar;
692
693   // 聯絡資訊
694   const emailLink = document.getElementById('user-email');
```

### index1.html, line 661 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890   function getCardDataFromNFC() {
891   // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892   const urlParams = new URLSearchParams(window.location.search);
893   let cardDataParam = urlParams.get('data');
894   let isCompactFormat = false;
895
```

#### Sink Details

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index1.html, line 661 (Cross-Site Scripting: DOM)

**Sink:** Assignment to avatar.src
**Enclosing Method:** renderCard()
**File:** index1.html:661
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
658
659  // 開始載入圖片
660  console.log('Loading avatar:', data.avatar);
661  avatar.src = data.avatar;
662
663  // 聯絡資訊
664  const emailLink = document.getElementById('user-email');
```

### index-en.html, line 702 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index-en.html:702
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
699  const phoneItem = document.getElementById('phone-item');
700  if (data.phone) {
701  const phoneLink = document.getElementById('user-phone');
702  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
703  phoneLink.textContent = data.phone;
704  phoneItem.style.display = 'flex';
705  } else {
```

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index-personal-en.html, line 637 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index-personal-en.html:637
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
634  const phoneItem = document.getElementById('phone-item');
635  if (data.phone) {
636  const phoneLink = document.getElementById('user-phone');
637  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
638  phoneLink.textContent = data.phone;
639  phoneItem.style.display = 'flex';
640  } else {
```

### index-personal.html, line 637 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784  }
785
786  function getCardDataFromNFC() {
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-personal.html, line 637 (Cross-Site Scripting: DOM)

```
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index-personal.html:637
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
634  const phoneItem = document.getElementById('phone-item');
635  if (data.phone) {
636  const phoneLink = document.getElementById('user-phone');
637  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
638  phoneLink.textContent = data.phone;
639  phoneItem.style.display = 'flex';
640  } else {
```

### index.html, line 672 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index.html:672
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
669  const phoneItem = document.getElementById('phone-item');
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index.html, line 672 (Cross-Site Scripting: DOM)

```
670  if (data.phone) {
671  const phoneLink = document.getElementById('user-phone');
672  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
673  phoneLink.textContent = data.phone;
674  phoneItem.style.display = 'flex';
675  } else {
```

### index1-en.html, line 702 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index1-en.html:702
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
699  const phoneItem = document.getElementById('phone-item');
700  if (data.phone) {
701  const phoneLink = document.getElementById('user-phone');
702  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
703  phoneLink.textContent = data.phone;
704  phoneItem.style.display = 'flex';
705  } else {
```

### index1.html, line 672 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index1.html, line 672 (Cross-Site Scripting: DOM)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to phoneLink.href
**Enclosing Method:** renderCard()
**File:** index1.html:672
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
669  const phoneItem = document.getElementById('phone-item');
670  if (data.phone) {
671  const phoneLink = document.getElementById('user-phone');
672  phoneLink.href = `tel:${data.phone.replace(/[^0-9+]/g, '')}`;
673  phoneLink.textContent = data.phone;
674  phoneItem.style.display = 'flex';
675  } else {
```

### index-en.html, line 695 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index-en.html, line 695 (Cross-Site Scripting: DOM)

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()
**File:** index-en.html:695
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
692
693  // 聯絡資訊
694  const emailLink = document.getElementById('user-email');
695  emailLink.href = data.socialLinks.email;
696  emailLink.textContent = data.email;
697
698  // 電話
```

### index-personal-en.html, line 631 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()
**File:** index-personal-en.html:631
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
628
629  // Contact information
630  const emailLink = document.getElementById('user-email');
631  emailLink.href = data.socialLinks.email;
632  emailLink.textContent = data.email;
633
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-personal-en.html, line 631 (Cross-Site Scripting: DOM)

```
634  const phoneItem = document.getElementById('phone-item');
```

### index-personal.html, line 631 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()
**File:** index-personal.html:631
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
628
629  // 聯絡資訊
630  const emailLink = document.getElementById('user-email');
631  emailLink.href = data.socialLinks.email;
632  emailLink.textContent = data.email;
633
634  const phoneItem = document.getElementById('phone-item');
```

### index.html, line 665 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

## Cross-Site Scripting: DOM — Critical

**Package: <none>**

### index.html, line 665 (Cross-Site Scripting: DOM)

```
889
890  function getCardDataFromNFC() {
891  // 從URL 參數中獲取NFC 傳送的JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()
**File:** index.html:665
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
662
663  // 聯絡資訊
664  const emailLink = document.getElementById('user-email');
665  emailLink.href = data.socialLinks.email;
666  emailLink.textContent = data.email;
667
668  // 電話（可選）
```

### index1-en.html, line 695 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index1-en.html, line 695 (Cross-Site Scripting: DOM)

**File:** index1-en.html:695
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
692
693  // 聯絡資訊
694  const emailLink = document.getElementById('user-email');
695  emailLink.href = data.socialLinks.email;
696  emailLink.textContent = data.email;
697
698  // 電話
```

### index1.html, line 665 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to emailLink.href
**Enclosing Method:** renderCard()
**File:** index1.html:665
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
662
663  // 聯絡資訊
664  const emailLink = document.getElementById('user-email');
665  emailLink.href = data.socialLinks.email;
666  emailLink.textContent = data.email;
667
668  // 電話（可選）
```

### index-en.html, line 713 (Cross-Site Scripting: DOM)

#### Issue Details

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-en.html, line 713 (Cross-Site Scripting: DOM)

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index-en.html:713
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
710  const mobileItem = document.getElementById('mobile-item');
711  if (data.mobile && data.mobile.trim() !== '') {
712  const mobileLink = document.getElementById('user-mobile');
713  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
714  mobileLink.textContent = data.mobile;
715  mobileItem.style.display = 'flex';
716  } else {
```

### index-personal-en.html, line 647 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-personal-en.html, line 647 (Cross-Site Scripting: DOM)

```
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index-personal-en.html:647
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
644  const mobileItem = document.getElementById('mobile-item');
645  if (data.mobile && data.mobile.trim() !== '') {
646  const mobileLink = document.getElementById('user-mobile');
647  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
648  mobileLink.textContent = data.mobile;
649  mobileItem.style.display = 'flex';
650  } else {
```

### index-personal.html, line 647 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index-personal.html:647
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
644  const mobileItem = document.getElementById('mobile-item');
645  if (data.mobile && data.mobile.trim() !== '') {
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index-personal.html, line 647 (Cross-Site Scripting: DOM)

```
646  const mobileLink = document.getElementById('user-mobile');
647  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
648  mobileLink.textContent = data.mobile;
649  mobileItem.style.display = 'flex';
650  } else {
```

### index.html, line 683 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index.html:683
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
680  const mobileItem = document.getElementById('mobile-item');
681  if (data.mobile && data.mobile.trim() !== '') {
682  const mobileLink = document.getElementById('user-mobile');
683  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
684  mobileLink.textContent = data.mobile;
685  mobileItem.style.display = 'flex';
686  } else {
```

### index1-en.html, line 713 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index1-en.html, line 713 (Cross-Site Scripting: DOM)

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index1-en.html:713
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
710  const mobileItem = document.getElementById('mobile-item');
711  if (data.mobile && data.mobile.trim() !== '') {
712  const mobileLink = document.getElementById('user-mobile');
713  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
714  mobileLink.textContent = data.mobile;
715  mobileItem.style.display = 'flex';
716  } else {
```

### index1.html, line 683 (Cross-Site Scripting: DOM)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

## Cross-Site Scripting: DOM — Critical

### Package: <none>

### index1.html, line 683 (Cross-Site Scripting: DOM)

#### Sink Details

**Sink:** Assignment to mobileLink.href
**Enclosing Method:** renderCard()
**File:** index1.html:683
**Taint Flags:** JS_OBJECT_CONTROLLED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
680  const mobileItem = document.getElementById('mobile-item');
681  if (data.mobile && data.mobile.trim() !== '') {
682  const mobileLink = document.getElementById('user-mobile');
683  mobileLink.href = `tel:${data.mobile.replace(/[^0-9+]/g, '')}`;
684  mobileLink.textContent = data.mobile;
685  mobileItem.style.display = 'flex';
686  } else {
```

# Cross-Site Scripting: Poor Validation (12 issues)

## Abstract

如果依賴 HTML、XML 和其他類型的編碼來驗證使用者輸入，可能會導致瀏覽器執行惡意程式碼。

## Explanation

使用特定編碼功能可防止部分，但不是所有的 Cross-Site Scripting 攻擊。根據資料出現的內容，以 HTML 編碼的基本 <、>、& 和 "，以及以 XML 編碼的 <、>、&、" 和 ' 以外的字元，可能會有中繼意義。依賴這類編碼功能等同於使用安全性較差的拒絕清單來防止 Cross-Site Scripting 攻擊，並且可能會允許攻擊者插入隨後會在瀏覽器中執行的惡意程式碼。因為並非隨時都可以準確識別其中靜態顯示資料的內容，所以即使套用編碼，並以 Cross-Site Scripting 來加以顯示，Fortify Secure Coding Rulepacks 還是會報告 Cross-Site Scripting 發現：Poor Validation 問題。 Cross-site scripting (XSS) 弱點會在以下情況中出現： 1.資料從一個不可信賴的來源進入 Web 應用程式。在基於 DOM 的 XSS 案例中，會從 URL 參數或瀏覽器內的其他值來讀取資料，並以用戶端程式碼回寫到頁面中。在 Reflected XSS 案例中，不可信賴的來源通常為網頁要求，而在 Persisted XSS (也可稱為 Stored XSS) 案例中，來源通常為資料庫或其他後端資料儲存區。 2.未經驗證且包含在動態內容中的資料將傳送給某個網頁使用者。在基於 DOM 的 XSS 案例中，一旦受害者的瀏覽器開始剖析 HTML 頁面，就會開始執行惡意內容，且成為建立 DOM (Document Object Model，文件物件模型) 的一部分。 傳送到網頁瀏覽器的惡意內容經常是以 JavaScript 片段的形式出現，但是也可能包含 HTML、Flash 或者瀏覽器執行的任何其他程式碼類型。以 XSS 為基礎的攻擊手段花樣百出且幾乎無窮無盡，但是它們通常會傳輸 Cookie 或其他階段作業資訊之類的私人資料給攻擊者、將受害者重新導向到攻擊者控制的網頁內容，或者利用易受攻擊的網站，在使用者的機器上執行其他惡意操作。 **範例 1：**以下 JavaScript 程式碼片段會從 HTTP 要求中讀取員工識別碼 `eid`，將識別碼去除，然後顯示給使用者。

```
<SCRIPT>
var pos=document.URL.indexOf("eid=")+4;
document.write(escape(document.URL.substring(pos,document.URL.length)));
</SCRIPT>
```

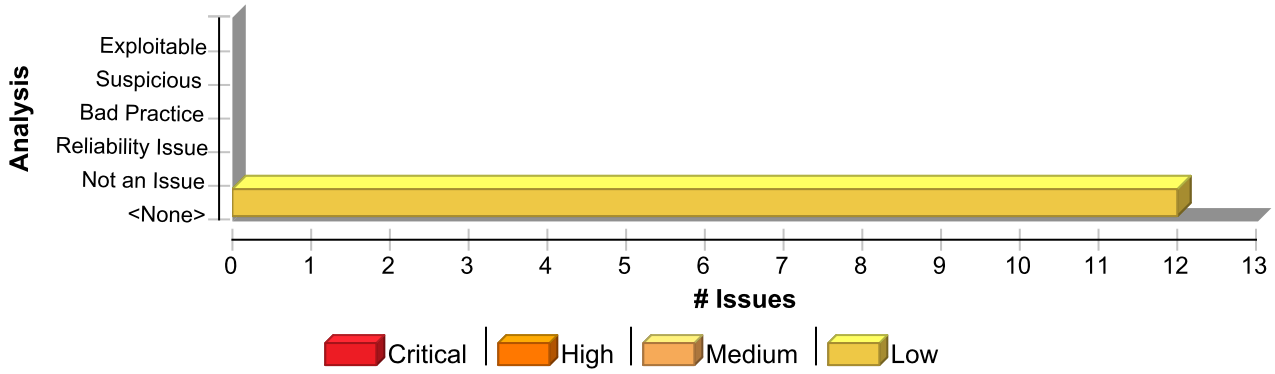
如果 `eid` 只包含標準英數字元，則這個範例中的程式碼會正確地執行。如果 `eid` 中有包含中繼字元或來源程式碼中的值，那麼網路瀏覽器就會像顯示 HTTP 回應那樣執行程式碼。 一開始，這似乎不會輕易受到攻擊。畢竟，誰會在自己的電腦中輸入會執行惡意程式碼的 URL？其實，真正的危險在於攻擊者會建立惡意的 URL，接著使用電子郵件或社交工程病毒誘騙受害者透過連結前往該 URL。當受害者按下該連結時，他們即不知不覺地透過易受攻擊的 Web 應用程式，將惡意內容資訊帶回他們自己的電腦。這個利用易受攻擊的 Web 應用程式進行攻擊的機制就是通常所知的 Reflected XSS。 如同範例中所示，XSS 的弱點是由 HTTP 回應中包含未經驗證資料的程式碼所引起的。XSS 攻擊有三種途徑可攻擊受害者： - 直接從 HTTP 要求讀取資料，並直接回傳至 HTTP 回應。當攻擊者誘使使用者提供危險內容給易受攻擊的 Web 應用程式，接著這些危險內容就會回傳給使用者並由在網路瀏覽器中執行，這時就會出現 Reflected XSS 攻擊行為。傳遞惡意內容最常用的機制就是，將惡意內容當作參數隱藏在公開發表的 URL 中，或者以電子郵件方式直接傳送給受害者。以這種方法建立的 URL 會構成許多網路釣魚 (phishing) 架構的核心，攻擊者可以藉此誘使受害者去造訪一個指向到易受攻擊網站的 URL。網站將攻擊者的內容回傳給使用者之後，就會執行這些內容，並接著從使用者的電腦將可能包含階段作業資訊的 Cookie 之類的私人資訊傳給攻擊者，或者執行其他惡意活動。 - 應用程式會將危險資料儲存到資料庫或其他可信任的資料儲存中。這些危險資料隨後會被回讀到應用程式，並包含在動態內容中。Persistent XSS 攻擊會在以下情況出現：攻擊者把危險內容插入到之後會讀取的資料記憶體中，並包含在動態內容中。從攻擊者的角度來看，插入惡意內容的最佳位置莫過於一個會對很多使用者或特別感興趣的使用者顯示的區域。感興趣的使用者通常會在應用程式中擁有較高的權限，或者會與敏感資料進行互動，且這些資料對攻擊者而言很有利用價值。如果其中一個使用者執行了惡意內容，攻擊者可能會代替使用者去執行需要權限許可的作業，或者取得存取使用者專屬敏感資料的權限。 - 應用程式以外的來源會在資料庫或是其他資料記憶體中儲存危險資料，且之後這些危險資料會被當作信賴的資料回讀到應用程式，並會包含在動態內容中。

## Recommendation

防止 XSS 的解決方法是：確保在需要的位置進行驗證，並設定相關屬性以防止漏洞。 由於 XSS 弱點會在應用程式於輸出中包含惡意資料時出現，所以合理方法就是，在應用程式輸出資料之前馬上驗證資料。然而，由於 Web 應用程式經常會為了產生動態內容而包含複雜且難以理解的程式碼，所以此方法容易遺漏錯誤 (遺

漏驗證)。降低此風險的有效方法即為執行 XSS 輸入驗證。 Web 應用程式必須驗證所有輸入以防止出現其他弱點 (如 SQL Injection)，因此比較簡單的方法就是，加強應用程式現有的輸入驗證機制，納入 XSS 檢查。儘管有一定的作用，但是 XSS 的輸入驗證並不能取代嚴格的輸出驗證。應用程式可能會透過共用資料存放區或者其他信賴的來源接受輸入，而該資料存放區可能會從未執行適當輸入驗證的來源接受輸入。因此，應用程式不能間接依賴此資料或任何其他資料的安全性。這代表杜絕 XSS 弱點的最佳方法就是，驗證所有進入應用程式的資料，以及所有離開應用程式並傳送到使用者的資料。 驗證 XSS 最安全的方法，是建立一個安全字元的允許清單，只允許清單上的字元可以出現在 HTTP 內容，並僅接受由經過檢驗的集合中字元組成的輸入。例如，有效的使用者名稱可能僅包含英數字元，或電話號碼可能僅包含數字 0-9。然而，這種解決方式在 Web 應用程式中經常是不可行的，因為很多字元對瀏覽器來說都具有特殊意義，將這些字元編碼之後，必須將它們視為有效輸入。例如，一個網站設計公告欄就必須接受來自於使用者的 HTML 片段。 更具彈性但安全性較低的方法是實作拒絕清單，能在使用輸入前選擇性地拒絕或避開可能有危險的字元。若要建立這類名單，首先必須了解那些對網頁瀏覽器來說有特殊意義的字元集。雖然 HTML 標準會定義具有特殊意義的字元，但是許多網頁瀏覽器會嘗試修正 HTML 中的常見錯誤，而且在特定環境中可能會將其他字元視為有特殊意義。這就是為什麼我們不建議使用拒絕清單做為預防 XSS 的方式。Carnegie Mellon 大學軟體工程學院 (Software Engineering Institute) 下的 CERT(R) 合作中心 (CERT(R) Coordination Center) 提供了在各種環境中具有特殊意義的字元資訊 [1]： 在區塊等級 (block-level) 元素的內容中 (位於文字段落中間)： -「<」是特殊字元，因為它會引入標籤。 -「&」是特殊字元，因為它會引入字元實體。 -「>」是特殊字元，因為某些瀏覽器將其視為特殊字元，會假設該頁面的作者想加入開頭的「<」，卻不小心遺漏掉了。 以下原則適用於屬性值： - 在以雙引號括住的屬性值中，雙引號之所以特殊，是因為它們標記了屬性值的結尾。 - 在以單引號結尾的屬性值中，單引號是特殊字元，因為它們標記了屬性值的結尾。 - 在沒有任何引號的屬性值中，空格字元 (例如空格和定位字元) 也是特殊字元。 -「&」和特定屬性一起使用時會變成特殊字元，因為它會引導出一個字元實體。 舉例來說，在 URL 中，搜索引擎可能會在結果頁面提供一個連結，讓使用者可以按一下該連結來重新執行搜尋。這個方法可以運用在編寫 URL 中的搜尋查詢，此動作會引導出其他特殊字元： - 空格、定位字元和換行符號都是特殊字元，因為它們標記了 URL 的結尾。 -「&」為特殊字元，因為它會引導出一個字元實體或分隔 CGI 參數。 - 非 ASCII 字元 (就是所有在 ISO-8859-1 編碼表中大於 127 的字元) 不允許出現在 URL 中，所以它們在此內容中被視為特殊字元。 - 每當伺服器端程式碼對以 HTTP 逸出序列編碼的參數進行解碼時，就必須從輸入中篩選 "%" 符號。例如，對於諸如「%68%65%6C%6C%6F」的輸入，必須篩選「%」，才能在網頁上顯示「hello」。 在 的正文中： - 將文字直接插入原有指令碼標籤時，必須篩選分號、括號、中括號以及換行字元。 伺服器端 Script： - 如果伺服器端 Script 會將輸入中的任何驚嘆號字元 (!) 轉換為輸出中的雙引號字元時 (")，則可能需要進行更多篩選。 其他可能性： - 若攻擊者提交 UTF-7 編碼的要求，特殊字元「<」會顯示為「+ADw-」，並且可能會避開篩選。如果輸出包含在沒有明確指定編碼格式的頁面中，某些瀏覽器會嘗試以智慧方式來根據內容識別編碼 (在此情況下為 UTF-7)。 在您確定在應用程式中針對 XSS 攻擊執行驗證的正確要點，以及驗證時要考慮的特殊字元後，那麼下一個挑戰就是決定驗證過程中處理特殊字元的方法。如果應用程式將某些特殊字元認定為無效輸入，那麼您可以拒絕任何包含這些被視為無效特殊字元的輸入。第二種選擇是以篩選方式移除特殊字元。然而，篩選所產生的副作用在於會改變篩選內容的顯示樣貌。但是在需要完整顯示輸入內容時，這種情況是無法接受的。 如果必須接受包含特殊字元的輸入且必須準確顯示輸入內容，驗證動作一定要編碼所有特殊字元，以移除特殊字元代表的意義。官方 HTML 規格 [2] 提供了特殊字元對應的 ISO 8859-1 編碼值完整清單。 很多應用程式伺服器都試圖避免應用程式出現 Cross-site scripting 弱點，方法是為負責設定某個特定 HTTP 回應內容的函數提供各種實作，以驗證是否存在進行 Cross-site scripting 攻擊的字元。請勿依賴執行應用程式的伺服器來確保應用程式的安全性。對於任何開發的應用程式，無法保證在其存留期內將在哪些應用程式伺服器上執行。隨著標準和已知盜取方式不斷地演變，無法保證應用程式伺服器會繼續保持同步。

**Issue Summary**




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| Cross-Site Scripting: Poor Validation | 12 | 0 | 0 | 12 |
| **Total** | **12** | **0** | **0** | **12** |

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index-en.html, line 863 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index-en.html:780

```
777  const timestamp = now.toISOString().replace(/[-:]/g, '').split('.')[0] + 'Z';
778
779  const currentOrigin = window.location.origin;
780  const currentPath = window.location.pathname.replace('/index-en.html', '').replace(/\/$/, '');
781  const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
782
783  let socialNote = '';
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-en.html:863
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index-en.html, line 863 (Cross-Site Scripting: Poor Validation)

```
860
861  const vcardContent = generateVCardContent();
862  const vcfData = encodeURIComponent(vcardContent);
863  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
864  contactLink.setAttribute('download', `${data.name}.vcf`);
865
866  contactLink.addEventListener('click', function() {
```

### index-personal-en.html, line 752 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index-personal-en.html:721

```
718  const timestamp = now.toISOString().replace(/[-:]/g, '').split('.')[0] +
'Z';
719
720  const currentOrigin = window.location.origin;
721  const currentPath = window.location.pathname.replace('/index-personal-
en.html', '').replace(/\/$/, '');
722  const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
723
724  let socialNote = '';
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-personal-en.html:752
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
749
750  const vcardContent = generateVCardContent();
751  const vcfData = encodeURIComponent(vcardContent);
752  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
753  contactLink.setAttribute('download', `${data.name}.vcf`);
754
755  contactLink.addEventListener('click', function() {
```

## Cross-Site Scripting: Poor Validation — Low

**Package: <none>**

**index-personal-en.html, line 752 (Cross-Site Scripting: Poor Validation)**

**index-personal.html, line 752 (Cross-Site Scripting: Poor Validation)**

### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index-personal.html:721

```
718  const timestamp = now.toISOString().replace(/[-:]/g, '').split('.')[0] + 'Z';
719
720  const currentOrigin = window.location.origin;
721  const currentPath = window.location.pathname.replace('/index-personal.html', '').replace(/\/$/, '');
722  const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
723
724  let socialNote = '';
```

### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-personal.html:752
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
749
750  const vcardContent = generateVCardContent();
751  const vcfData = encodeURIComponent(vcardContent);
752  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
753  contactLink.setAttribute('download', `${data.name}.vcf`);
754
755  contactLink.addEventListener('click', function() {
```

**index.html, line 855 (Cross-Site Scripting: Poor Validation)**

### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index.html:760

```
757
758   // 動態生成照片 URL（基於當前網站位置）
759   const currentOrigin = window.location.origin;
760   const currentPath = window.location.pathname.replace('/index.html',
'').replace(/\/$/, '');
761   const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
762
763   // 準備社群媒體備註
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index.html:855
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
852   // 設定下載連結
853   const vcardContent = generateVCardContent();
854   const vcfData = encodeURIComponent(vcardContent);
855   contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
856   contactLink.setAttribute('download', `${data.name}.vcf`);
857
858   // 監聽按鈕點擊事件
```

### index1-en.html, line 863 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index1-en.html:780

```
777   const timestamp = now.toISOString().replace(/[-:]/g, '').split('.')[0] +
'Z';
778
```

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index1-en.html, line 863 (Cross-Site Scripting: Poor Validation)

```
779  const currentOrigin = window.location.origin;
780  const currentPath = window.location.pathname.replace('/index1-en.html', '').replace(/\/$/, '');
781  const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
782
783  let socialNote = '';
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index1-en.html:863
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
860
861  const vcardContent = generateVCardContent();
862  const vcfData = encodeURIComponent(vcardContent);
863  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
864  contactLink.setAttribute('download', `${data.name}.vcf`);
865
866  contactLink.addEventListener('click', function() {
```

### index1.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.pathname
**From:** generateVCardContent
**File:** index1.html:760

```
757
758  // 動態生成照片 URL（基於當前網站位置）
759  const currentOrigin = window.location.origin;
760  const currentPath = window.location.pathname.replace('/index1.html', '').replace(/\/$/, '');
761  const photoUrl = `${currentOrigin}${currentPath}/${data.avatar}`;
762
763  // 準備社群媒體備註
```

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index1.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index1.html:855
**Taint Flags:** CONCATENATED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
852  // 設定下載連結
853  const vcardContent = generateVCardContent();
854  const vcfData = encodeURIComponent(vcardContent);
855  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
856  contactLink.setAttribute('download', `${data.name}.vcf`);
857
858  // 監聽按鈕點擊事件
```

### index-en.html, line 863 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-en.html:863
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT,

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index-en.html, line 863 (Cross-Site Scripting: Poor Validation)

VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
860
861  const vcardContent = generateVCardContent();
862  const vcfData = encodeURIComponent(vcardContent);
863  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
864  contactLink.setAttribute('download', `${data.name}.vcf`);
865
866  contactLink.addEventListener('click', function() {
```

### index-personal-en.html, line 752 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal-en.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-personal-en.html:752
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
749
750  const vcardContent = generateVCardContent();
751  const vcfData = encodeURIComponent(vcardContent);
752  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
753  contactLink.setAttribute('download', `${data.name}.vcf`);
754
```

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index-personal-en.html, line 752 (Cross-Site Scripting: Poor Validation)

```
755  contactLink.addEventListener('click', function() {
```

### index-personal.html, line 752 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index-personal.html:787

```
784  }
785
786  function getCardDataFromNFC() {
787  const urlParams = new URLSearchParams(window.location.search);
788  let cardDataParam = urlParams.get('data');
789  let isCompactFormat = false;
790
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index-personal.html:752
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEAKCRYPTO, WEB, XSS

```
749
750  const vcardContent = generateVCardContent();
751  const vcfData = encodeURIComponent(vcardContent);
752  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
753  contactLink.setAttribute('download', `${data.name}.vcf`);
754
755  contactLink.addEventListener('click', function() {
```

### index.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從 URL 參數中獲取 NFC 傳送的 JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index.html:855
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
852  // 設定下載連結
853  const vcardContent = generateVCardContent();
854  const vcfData = encodeURIComponent(vcardContent);
855  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
856  contactLink.setAttribute('download', `${data.name}.vcf`);
857
858  // 監聽按鈕點擊事件
```

### index1-en.html, line 863 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1-en.html:898

```
895  }
896
897  function getCardDataFromNFC() {
```

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index1-en.html, line 863 (Cross-Site Scripting: Poor Validation)

```
898  const urlParams = new URLSearchParams(window.location.search);
899  let cardDataParam = urlParams.get('data');
900  let isCompactFormat = false;
901
```

#### Sink Details

**Sink:** Assignment to contactLink.href
**Enclosing Method:** setupVCardLink()
**File:** index1-en.html:863
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
860
861  const vcardContent = generateVCardContent();
862  const vcfData = encodeURIComponent(vcardContent);
863  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
864  contactLink.setAttribute('download', `${data.name}.vcf`);
865
866  contactLink.addEventListener('click', function() {
```

### index1.html, line 855 (Cross-Site Scripting: Poor Validation)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read window.location.search
**From:** getCardDataFromNFC
**File:** index1.html:892

```
889
890  function getCardDataFromNFC() {
891  // 從URL 參數中獲取NFC 傳送的JSON 資料
892  const urlParams = new URLSearchParams(window.location.search);
893  let cardDataParam = urlParams.get('data');
894  let isCompactFormat = false;
895
```

#### Sink Details

**Sink:** Assignment to contactLink.href

## Cross-Site Scripting: Poor Validation — Low

### Package: <none>

### index1.html, line 855 (Cross-Site Scripting: Poor Validation)

**Enclosing Method:** setupVCardLink()
**File:** index1.html:855
**Taint Flags:** CONCATENATED, JS_OBJECT_CONTROLLED, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_AI, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_DOM_AI, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_OPEN_REDIRECT, WEB, XSS

```
852  // 設定下載連結
853  const vcardContent = generateVCardContent();
854  const vcfData = encodeURIComponent(vcardContent);
855  contactLink.href = `data:text/vcard;charset=utf-8,${vcfData}`;
856  contactLink.setAttribute('download', `${data.name}.vcf`);
857
858  // 監聽按鈕點擊事件
```

## Cross-Site Scripting: Self (2 issues)

### Abstract

傳送未經驗證的資料至網路瀏覽器，會導致瀏覽器執行惡意的程式碼。

### Explanation

Cross-Site Scripting (XSS) 弱點會在以下情況中出現： 1.資料從一個不可信賴的來源進入 Web 應用程式。在自我 XSS 的案例中，會從文字方塊或可從 DOM 控制的其他值來讀取資料，並以用戶端程式碼回寫到頁面中。 2.未經驗證且包含在動態內容中的資料將傳送給某個網頁使用者。在自我 XSS 的案例中，惡意內容會做為 DOM (Document Object Model，文件物件模型) 修改的一部分執行。 在自我 XSS 的案例中，惡意內容是以 JavaScript 片段，或瀏覽器執行的其他程式碼類型的形式出現。由於自我 XSS 主要是對自己的攻擊，所以常被認為不重要，但如果可能發生下列任一情況，則應視為標準 XSS 弱點來處理： - 在您的網站上識別出「跨網站偽造要求」弱點。 - 社交工程攻擊可能說服某個使用者攻擊自己的帳戶，進而危及其階段作業。 **範例 1：**請考慮使用 HTML 表單：

```
<div id="myDiv">
Employee ID: <input type="text" id="eid"><br>
...
<button>Show results</button>
</div>
<div id="resultsDiv">
...
</div>
```

以下的 jQuery 程式碼片段會從文字方塊讀取員工識別碼，並顯示給使用者。

```
$(document).ready(function(){
$("#myDiv").on("click", "button", function(){
var eid = $("#eid").val();
$("resultsDiv").append(eid);
...
});
});
```

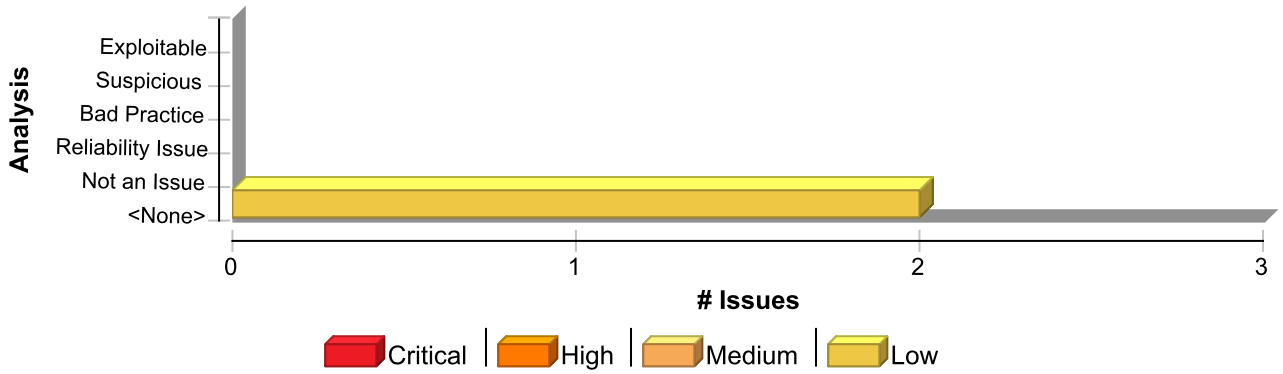
如果文字輸入中的員工識別碼 (識別碼為 eid) 只包含標準英數字元，這些程式碼便會正確地運作。如果 eid 中有包含中繼或來源程式碼中的值，那麼在使用者按一下按鈕後，程式碼就會被新增至 DOM 供瀏覽器執行。如果攻擊者可以說動使用者將惡意內容輸入文字輸入中，那麼這就只是 DOM 型 XSS。

### Recommendation

防止 XSS 的解決方法是：確保在需要的位置進行驗證，並設定相關屬性以防止漏洞。 由於 XSS 弱點會在應用程式在它的輸出中包含惡意資料時出現，所以合理方法就是，在資料離開應用程式之前馬上驗證資料 (如果是 DOM 型，會在轉譯之前馬上驗證)。然而，由於 Web 應用程式經常會為了產生動態內容而包含複雜且難以理解的程式碼，所以此方法容易遺漏錯誤 (遺漏驗證)。降低此風險的有效方法即為執行 XSS 輸入驗證。 Web 應用程式必須驗證所有輸入以防止出現其他弱點 (如 SQL Injection)，因此比較簡單的方法就是，加強應用程式現有的輸入驗證機制，納入 XSS 檢查。儘管有一定的作用，但是 XSS 的輸入驗證並不能取代嚴格的輸出驗證。應用程式可能會透過共用資料存放區或者其他信賴的來源接受輸入，而該資料存放區可能會從未執行適當輸入驗證的來源接受輸入。因此，應用程式不能間接依賴此資料或任何其他資料的安全性。這代表杜絕 XSS 弱點的最佳方法就是，驗證所有進入應用程式的資料，以及所有離開應用程式並傳送到使用者的資料。 驗證 XSS 最安全的方法，是建立一個安全字元的允許清單，只允許清單上的字元可以出現在 HTTP 內容，並僅接受由經過檢驗的集合中字元組成的輸入。例如，有效的使用者名稱可能僅包含英數字元，或電話號碼可能僅包含數字 0-9。然而，這種解決方式在 Web 應用程式中經常是不可行的，因為很多字元對瀏覽器來說都具有特殊意義，將這些字元編碼之後，必須將它們視為有效輸入。例如，一個網站設計公告欄就必須接受來自於使用者的 HTML 片段。 更具彈性但安全性較低的方法是實作拒絕清單，能在使用輸入前選擇性地拒絕或避開可能有危險的字元。若要建立這類名單，首先必須了解那些對網頁瀏覽器來說有特殊意義的字元集。雖然 HTML 標準會定義具有特殊意義的字元，但是許多網頁瀏覽器會嘗試修正 HTML 中的常見錯誤，而且在特定環境中可能會將其他字元視為有特殊意義。這就是為什麼我們不建議使用拒絕清單做為預防 XSS 的方式。Carnegie Mellon 大學軟體工程學院 (Software Engineering Institute) 下的 CERT(R) 合作中心

(CERT(R) Coordination Center) 提供了在各種環境中具有特殊意義的字元資訊 [1]： 在區塊等級元素的內容中 (位於文字段落中間)： -「<」是特殊字元，因為它會引入標籤。 -「&」是特殊字元，因為它會引入字元實體。 -「>」是特殊字元，因為某些瀏覽器將其視為特殊字元，會假設該頁面的作者想加入開頭的「<」，卻不小心遺漏掉了。 以下原則適用於屬性值： - 在以雙引號括住的屬性值中，雙引號之所以特殊，是因為它們標記了屬性值的結尾。 - 在以單引號結尾的屬性值中，單引號是特殊字元，因為它們標記了屬性值的結尾。 - 在沒有任何引號的屬性值中，空格字元 (例如空格和定位字元) 也是特殊字元。 -「&」和特定屬性一起使用時會變成特殊字元，因為它會引入字元實體。 舉例來說，在 URL 中，搜索引擎可能會在結果頁面提供一個連結，讓使用者可以按一下該連結來重新執行搜尋。這個方法可以運用在編寫 URL 中的搜尋查詢，此動作會引導出其他特殊字元： - 空格、定位字元和換行符號都是特殊字元，因為它們標記了 URL 的結尾。 -「&」為特殊字元，因為它會引入字元實體或分隔 CGI 參數。 - 非 ASCII 字元 (就是所有在 ISO-8859-1 編碼表中大於 127 的字元) 不允許出現在 URL 中，所以它們在此內容中被視為特殊字元。 - 每當伺服器端程式碼對以 HTTP 逸出序列編碼的參數進行解碼時，就必須從輸入中篩選 "%" 符號。例如，對於諸如「%68%65%6C%6C%6F」的輸入，必須篩選「%」，才能在網頁上顯示「hello」。 在 的正文中： - 將文字直接插入原有指令碼標籤時，必須篩選分號、括號、中括號以及換行字元。 伺服器端 Script： - 如果伺服器端指令碼將輸入中的任何驚嘆號字元 (!) 轉換為輸出中的雙引號字元 (")，則可能需要進行更多篩選。 其他可能性： - 若攻擊者提交 UTF-7 的要求，特殊字元「<」會顯示為「+ADw-」，並且可能會避開篩選。如果輸出包含在沒有明確指定編碼格式的頁面中，某些瀏覽器會嘗試以智慧方式來根據內容識別編碼 (在此情況下為 UTF-7)。 在您確定在應用程式中針對 XSS 攻擊執行驗證的正確要點，以及驗證時要考慮的特殊字元後，那麼下一個挑戰就是決定驗證過程中處理特殊字元的方法。如果應用程式將某些特殊字元認定為無效輸入，那麼您可以拒絕任何包含這些被視為無效特殊字元的輸入。第二種選擇是以篩選方式移除特殊字元。然而，篩選所產生的副作用在於會改變篩選內容的顯示樣貌。但是在需要完整顯示輸入內容時，這種情況是無法接受的。 如果必須接受包含特殊字元的輸入，並將其準確顯示出來，驗證動作一定要編碼所有特殊字元，以移除特殊字元代表的意義。官方 HTML 規格 [2] 提供了特殊字元對應的 ISO 8859-1 編碼值完整清單。 很多應用程式伺服器都試圖避免應用程式出現 Cross-site scripting 弱點，方法是為負責設定某個特定 HTTP 回應內容的函數提供各種實作，以驗證是否存在進行 Cross-site scripting 攻擊的字元。請勿依賴執行應用程式的伺服器來確保應用程式的安全性。對於任何開發的應用程式，無法保證在其存留期內將在哪些應用程式伺服器上執行。隨著標準和已知盜取方式不斷地演變，無法保證應用程式伺服器會繼續保持同步。

## Issue Summary




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| Cross-Site Scripting: Self | 2 | 0 | 0 | 2 |
| **Total** | **2** | **0** | **0** | **2** |

**Cross-Site Scripting: Self** — **Low**

**Package: <none>**

**nfc-generator-bilingual.html, line 507 (Cross-Site Scripting: Self)**

**Issue Details**

## Cross-Site Scripting: Self — Low

### Package: <none>

### nfc-generator-bilingual.html, line 507 (Cross-Site Scripting: Self)

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read value
**From:** updateSocialPreview
**File:** nfc-generator-bilingual.html:498

```
495
496   // 社群連結預覽功能
497   function updateSocialPreview() {
498   const socialNote = document.getElementById('socialNote').value.trim();
499   const previewElement = document.getElementById('socialPreview');
500   const previewContent = document.getElementById('socialPreviewContent');
501
```

#### Sink Details

**Sink:** Assignment to previewContent.innerHTML
**Enclosing Method:** updateSocialPreview()
**File:** nfc-generator-bilingual.html:507
**Taint Flags:** CONCATENATED, SELF_XSS, WEB

```
504   if (socialNote) {
505   // 使用簡化版的社群連結處理
506   const processedLinks = processSocialLinksPreview(socialNote);
507   previewContent.innerHTML = processedLinks;
508   previewElement.style.display = 'block';
509   } else {
510   previewElement.style.display = 'none';
```

### Package: assets

### assets/bilingual-common.js, line 535 (Cross-Site Scripting: Self)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read value
**From:** processTest
**File:** test-social-links.html:153

```
150
151   <script>
152   function processTest() {
```

## Cross-Site Scripting: Self — Low

### Package: assets

### assets/bilingual-common.js, line 535 (Cross-Site Scripting: Self)

```
153  const input = document.getElementById('inputText').value;
154  const output = document.getElementById('outputArea');
155
156  if (!input.trim()) {
```

### Sink Details

**Sink:** Assignment to link.href
**Enclosing Method:** createSocialElement()
**File:** assets/bilingual-common.js:535
**Taint Flags:** CONCATENATED, SELF_XSS, WEB

```
532  label.textContent = platform + (displayUrl ? `: ${displayUrl}` : '');
533
534  const link = document.createElement('a');
535  link.href = url;
536  link.target = '_blank';
537  link.classNam = 'social-link';
538  link.style.cssText = `background: ${brandColor}; color: white; padding: 4px 12px; border-radius: 16px; text-decoration: none; font-size: 0.85em; font-weight: 500;`;
```

## Open Redirect (1 issue)

### Abstract

允許未經驗證的輸入控制重新導向中的 URL，有助於網路釣魚攻擊。

### Explanation

重新導向會在相同 Web 應用程式中將使用者導引到不同的網頁，或導引到外部網站。應用程式會利用重新導向協助站點內的導覽，且在某些情況下，會追蹤使用者離開網站的方法。Web 應用程式將用戶端重新導向至可由攻擊控制的任何任意 URL 時，會發生開放式重新導向弱點。攻擊者可能利用 Open Redirect 誘騙使用者造訪可信任網站的 URL，但接著會將其重新導向至惡意網站。透過對 URL 進行編碼，攻擊者可讓一般使用者難以注意到重新導向的惡意目標，即使將這個目標當作 URL 參數傳遞到受信任的網站也是如此。Open Redirect 通常為網路釣魚所使用的方法之一，可用來取得敏感的一般使用者資料。**範例 1：**當使用者按一下連結時，以下 JavaScript 程式碼會指示使用者的瀏覽器開啟由 `dest` 要求參數所讀取的 URL。

```
...
strDest = form.dest.value;
window.open(strDest,"myresults");
...
```

如果受害者收到一封電子郵件，指示他們開啟連結「http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com」，則該使用者可能會按下此連結，並認為自己會前往一個可信任的網站。不過，當受害者按下連結時，`Example 1` 中的程式碼會將瀏覽器重新導向至「http://www.wilyhacker.com」。許多使用者一直以來都被告知，要檢查在電子郵件中所接收到的 URL，以確定該連結是指向他們知悉的可信任網站。不過，如果攻擊者使用 Hex 以下列方式編碼目標的 URL： "http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D" 那麼，即使再有經驗的終端使用者也可能會被誘騙進入以下連結。

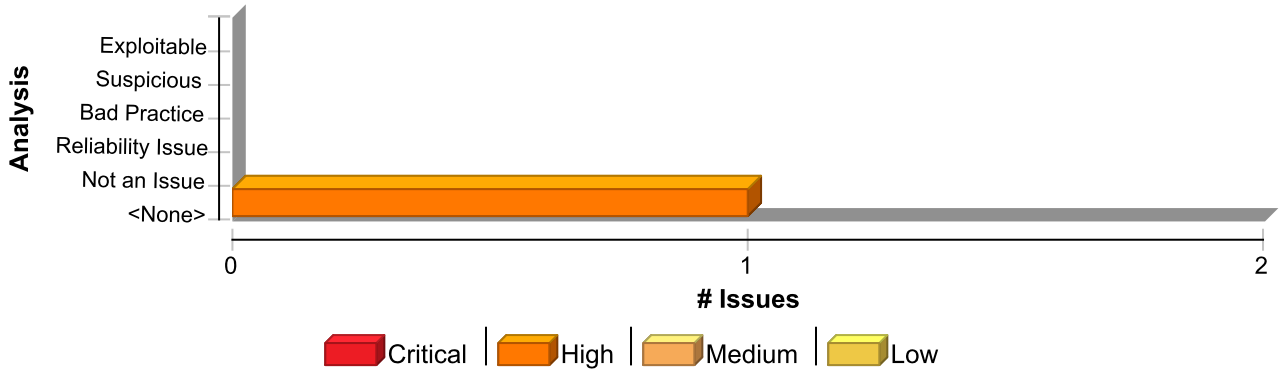
### Recommendation

不應允許未經驗證的使用者輸入，以重新導向的方式來控制目標的 URL。請改為使用間接方法：建立允許使用者指定的合法 URL 清單，且只允許使用者從該清單中進行選擇。使用這種方法，使用者所提供的輸入就不會直接被用來指定某個 URL 來進行重新導向。**範例 2：**以下程式碼會參照填入有效 URL 的陣列。使用者所按下的連結會傳送到與所要 URL 相對應的陣列索引中。

```
...
strDest = form.dest.value;
if((strDest.value != null)||(strDest.value.length!=0))
{
if((strDest >= 0) && (strDest <= strURLArray.length -1 ))
{
strFinalURL = strURLArray[strDest];
window.open(strFinalURL,"myresults");
}
}
...
```

但在某些情況下，這種方法是不切實際的，因為合法的 URL 太過於龐大或是難以追蹤。在這種情況下，請使用類似的方法來限制可重新導向使用者的網域，這麼一來，至少可以避免攻擊者將使用者引導到惡意的外部網站。

### Issue Summary




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| Open Redirect | 1 | 0 | 0 | 1 |
| **Total** | **1** | **0** | **0** | **1** |

## Open Redirect — High

### Package: assets

### assets/bilingual-common.js, line 535 (Open Redirect)

#### Issue Details

**Kingdom:** Input Validation and Representation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** Read value
**From:** processTest
**File:** test-social-links.html:153

```
150
151  <script>
152  function processTest() {
153  const input = document.getElementById('inputText').value;
154  const output = document.getElementById('outputArea');
155
156  if (!input.trim()) {
```

#### Sink Details

**Sink:** Assignment to link.href
**Enclosing Method:** createSocialElement()
**File:** assets/bilingual-common.js:535
**Taint Flags:** CONCATENATED, SELF_XSS, WEB

```
532  label.textContent = platform + (displayUrl ? `: ${displayUrl}` : '');
533
534  const link = document.createElement('a');
535  link.href = url;
536  link.target = '_blank';
```

## Open Redirect — High

### Package: assets

### assets/bilingual-common.js, line 535 (Open Redirect)

```
537  link.className = 'social-link';
538  link.style.cssText = `background: ${brandColor}; color: white; padding: 4px 12px; border-radius: 16px; text-decoration: none; font-size: 0.85em; font-weight: 500;`;
```

# System Information Leak: Internal (1 issue)

## Abstract

顯示系統資料或除錯資訊可讓攻擊者使用系統資訊制訂攻擊計畫。

## Explanation

在透過列印或記錄將系統資料或除錯資訊傳送至本機檔案、主控台或螢幕時，會發生內部資訊洩漏。 **範例 1：**以下程式碼會將異常寫入標準錯誤串流：

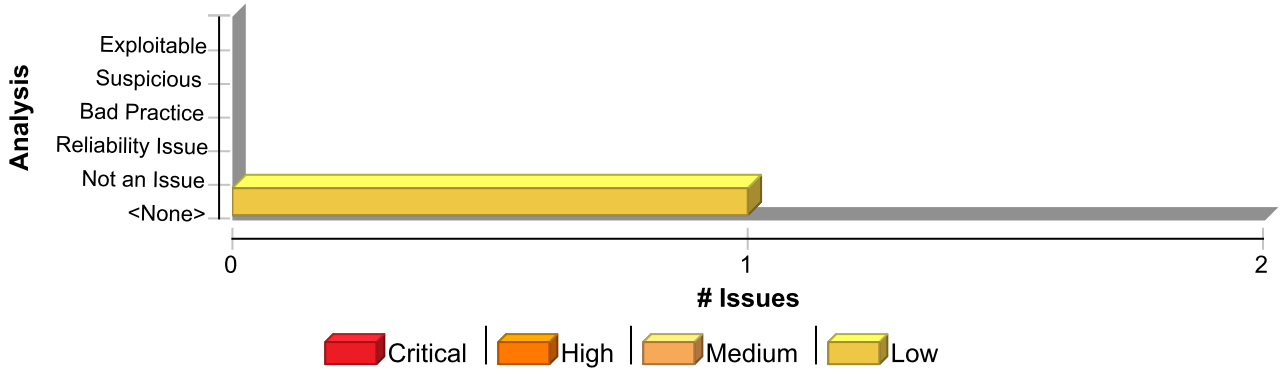
```
var http = require('http');
...

http.request(options, function(res){
...
}).on('error', function(e){
console.log('There was a problem with the request: ' + e);
});
...
```

根據系統配置，可以將資訊傾印到主控台，寫入記錄檔，或者顯示給使用者。在某些情況下，錯誤訊息會為攻擊者提供系統容易受到哪些確切的攻擊類型影響。例如，資料庫錯誤訊息即可揭露應用程式容易受到 SQL injection 攻擊。其他錯誤訊息還可揭露更多關於系統的間接線索。在 `Example 1` 中，洩漏的資訊可能會暗示有關作業系統類型、系統上安裝的應用程式，以及管理員在配置程式時所花費的努力等資訊。

## Recommendation

編寫錯誤訊息時，請將安全性問題考慮進去。在生產環境中，請儘量使用簡短的錯誤訊息，而不要使用詳細的錯誤資訊。限制產生與儲存詳細的輸出內容，將有助於管理員和程式設計師診斷問題。除錯追蹤有時可能會出現在不明顯的地方 (例如，內嵌在錯誤頁面的 HTML 註解中)。即使是不會揭露堆疊追蹤或資料庫傾印的簡短錯誤訊息，都可能會助攻擊者一臂之力。例如，「拒絕存取」訊息可能表示系統中存在著檔案或使用者。

## Issue Summary




## Engine Breakdown

| | SCA | WebInspect | SecurityScope | Total |
|---|---|---|---|---|
| System Information Leak: Internal | 1 | 0 | 0 | 1 |
| **Total** | **1** | **0** | **0** | **1** |

## System Information Leak: Internal — Low

### Package: <none>

### nfc-generator.html, line 676 (System Information Leak: Internal)

#### Issue Details

**Kingdom:** Encapsulation
**Scan Engine:** SCA (Data Flow)

#### Source Details

**Source:** lambda(0)
**From:** lambda
**File:** nfc-generator.html:675

```
672   btn.textContent = originalText;
673   btn.style.background = '';
674   }, 2000);
675   }).catch(function(err) {
676   console.error('複製失敗:', err);
677   alert('複製失敗，請手動複製連結');
678   });
```

#### Sink Details

**Sink:** ~JS_Generic.error()
**Enclosing Method:** lambda()
**File:** nfc-generator.html:676
**Taint Flags:** SYSTEMINFO

```
673   btn.style.background = '';
674   }, 2000);
675   }).catch(function(err) {
676   console.error('複製失敗:', err);
677   alert('複製失敗，請手動複製連結');
678   });
679   }
```

# Description of Key Terminology

## Likelihood and Impact

**Likelihood**
Likelihood is the probability that a vulnerability will be accurately identified and successfully exploited.

**Impact**
Impact is the potential damage an attacker could do to assets by successfully exploiting a vulnerability. This damage can be in the form of, but not limited to, financial loss, compliance violation, loss of brand reputation, and negative publicity.

## Fortify Priority Order

**Critical**
Critical-priority issues have high impact and high likelihood. Critical-priority issues are easy to detect and exploit and result in large asset damage. These issues represent the highest security risk to the application. As such, they should be remediated immediately.

SQL Injection is an example of a critical issue.

**High**
High-priority issues have high impact and low likelihood. High-priority issues are often difficult to detect and exploit, but can result in large asset damage. These issues represent a high security risk to the application. High-priority issues should be remediated in the next scheduled patch release.

Password Management: Hardcoded Password is an example of a high issue.

**Medium**
Medium-priority issues have low impact and high likelihood. Medium-priority issues are easy to detect and exploit, but typically result in small asset damage. These issues represent a moderate security risk to the application. Medium-priority issues should be remediated in the next scheduled product update.

Path Manipulation is an example of a medium issue.

**Low**
Low-priority issues have low impact and low likelihood. Low-priority issues can be difficult to detect and exploit and typically result in small asset damage. These issues represent a minor security risk to the application. Low-priority issues should be remediated as time allows.

Dead Code is an example of a low issue.

# About Fortify Solutions

Fortify is the leader in end-to-end application security solutions with the flexibility of testing on-premise and on-demand to cover the entire software development lifecycle. Learn more at www.microfocus.com/solutions/application-security.