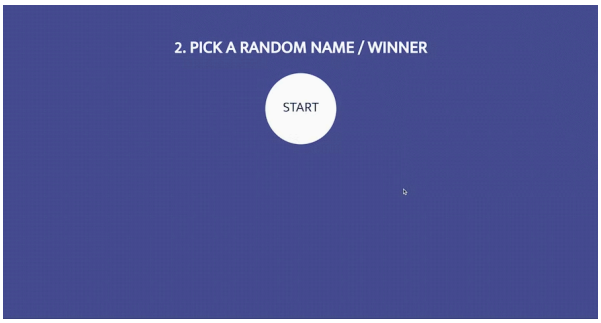
How our community hacked our own XSS challenge

BY INTIGRITI · MAY 27, 2019 · LAST UPDATED ON MARCH 6, 2025

Following the succes of our first XSS challenge, we decided to treat our Twitter followers for another game of "spot the cross-site scripting vulnerability":

> "NEW CHALLENGE: We're giving away a Burp Pro license, swag & invites to celebrate 5k followers! Claim your prize: https://t.co/KYQHSOpGvn #BugBounty #CTF #HackWithIntigriti pic.twitter.com/h2jDTM3qos
>
> — Intigriti (@intigriti) May 21, 2019"

The rules of the game were simple: identify a way to execute arbitrary javascript through the code and submit your solution through our bug bounty platform. From all 114 valid submissions, we drew one winner that gets a Burp Pro License, exclusive swag and a bunch of private Bug Bounty program invitations:



Newly joined intigriti community member Bincup has got luck on his side and got randomly picked as the winner of the challenge. Congratulations, Bincup!

The solution(s)

A challenge like this is like a box of chocolates: you never know what you're gonna get. While we created this challenge with one intended solution, our community found three ways to do the job. This once again highlights the strength in numbers that define bug bounty platforms: the more brains that look at your code, the more bugs might come out.

After the challenge was over, we encouraged people to share their solutions online so others could learn from them. We've listed some of the best writeups we've seen so far below:

@dee__see injected his payload in a malformed content-type header:

> "Writeup for @intigriti XSS Challenge 2! https://t.co/i6SvT4Jign Same recipe as the previous writeup + details on how I lost a ton of time on a bad assumption. pic.twitter.com/MvijNQz6gU
>
> — Dominic (@dee__see) May 26, 2019"

@hasp0t and @GeneralEG64 embedded their payloads in a base64 string in the URL:

> "Solution of the latest @intigriti challenge. Feedback appreciated!https://t.co/Rw1GP9EdOH
>
> — hasp0t (@hasp0t) May 25, 2019"

> "Intigriti – 5k followers #challenge solution!!
>
> Challenge ended, check out the source: https://t.co/5asFTQUGSN@intigriti @TipsBug @XssPayloads#GeneralEG pic.twitter.com/gR1EcfWgJr
>
> — Youssef A. Mohamed (@GeneralEG64) May 25, 2019"

Last but not least, @karouf went the extra mile and found both solutions:

> **"Ok so now that @intigriti #xss #challenge is over, please tell me what the damn official solution is! For the record, here's the one I found: https://t.co/DERsTg2lEz #ctf**
>
> **— Renaud Martinet (@karouf) May 24, 2019"**

A third and unexpected solution came from **0xNear** and **dkasak**, who found the only self-containing solution without the use of any external resources:

> **"https://challenge.intigriti.io/2/index.html;/#lj48c2NyaXB0PmFsZXJ0KGRvY3VtZW50LmRvbWFpbik8L3NjcmlwdD4="**

Granted, we didn't expect this one coming. It seems like our server was treating semicolons as a delimiter according to the obsolete RFC2396. This resulted in the delimited part of the URL, containing the base64 string, being ignored in the XHR request, causing a 200 OK status code, bypassing the check that verifies whether the image exists.

Last but not least, the most creative solution was raised by @mathias, who proposed to buy a TLD (like .com or .me) instead of using decimal or hexadecimal encoded IP addresses. Technically, there is nothing that withholds a TLD owner to host files at the root domain. While this wouldn't be the cheapest or fastest solution, it's *is* the shortest one:

http://challenge.intigriti.io/2/#//me

For those who don't like to read, YouTuber @GoatSniff released a video walkthrough that provides an excellent step-by-step tutorial in how to solve challenges like these:

The tips

Five tips were tweeted during this challenge:

1. There are two solutions! (this turned out to be wrong)
2. Try the unsecure version (for the first and second solution to work, you'd need to load the challenge over *http* instead of *https*)
3. Hexternal resources could help you. (the first and second solution required an external IP address, in their hexadecimal or decimal notation)
4. Read the specs and add some padding (in order to get the base64 payload to work, you sometimes had to add some padding to ensure proper decoding)
5. ;, hinting at the unintended solutions by **0xNear** and **dkasak**

Thank you!

Thanks to the community for participating in the challenge and congratulations to the 90 researchers who solved the challenge. A shout-out to the winner **Bincup**, who won a Burp License, swag package and private invites on our platform.

Didn't win this time? Keep your head up! We'll be launching more similar challenges on our Twitter account soon.

Got a cool idea for a challenge? Let us know at letstalk@intigriti.com and we might feature your idea in one of our upcoming challenges!

REQUEST A DEMO

intigriti.com/demo

VISIT THE WEBSITE

intigriti.com

GET IN TOUCH

hello@intigriti.com