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

Product: Whatsapp Messenger
Platforms: Web
Vulnerability type: XSS
Title: WHATSAPP - One Click Persistent XSS 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).

In this case, there are a few filtering bypasses that had to be done.
The first filter bypass is based on the same concept as of the previous report A.K.A "**One Click Open Redirect via URL filtering bypass using at sign (@) [MEDIUM RISK]**" on Whatsapp.
That is, intercepting the new message request and replacing the legitimate preview banner URL (e.g. https://facebook.com) with a malicious XSS URL(e.g. "javascript:alert(123);")

However, this is not enough to bypass Whatsapp's URL filtering mechanism, since in order for it not to omit the preview banner, it also expects the message to actually contain a legitimate url. Therefore "javascript:alert(123);" won't cut it.
This can also be bypassed by simply include a legitimate url in the new malformed url, thus tricking whatsapp into allowing the preview banner to stay (e.g. "javascript:'https://facebook.com'; alert(123);")

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.

There is a very serious risk in this vulnerability being open

Impact

HIGH RISK impact - using this, an attacker can send a malicious message to a victim, that once is clicked can bring to a full XSS on the victim's web.whatsapp.com context.

NOTE: in order to perform this attack, the payload has to exist within the body of the message (this is one filter I was not able to bypass)
To improve vector of attack, the attacker may locate the payload at the very bottom of the message, thus hiding it where it can only be seen after clicking "Read more" as we will see in the following example

With this, an attacker can achieve anything:

- stealing: cookies, messages, conversations, pictures, etc
- taking actions on behalf of the victim: delete messages, send messages, spread the XSS on to other users
- Anything really

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

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)

Browser:

- Theoretically: Agnostic [ALL] - since the XSS exists within the web app and not the browser
- Practically: just recently chromium has implemented a blocking mechanism for redirection to “javascript:” urls. Up until Chrome 75 it worked literally on every chrome based browser. But since that update, the vulnerability can only be exploited on Safari and Edge (before it went chromium). Firefox also blocks this behaviour.

OS: Agnostic [ALL]

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/B6UH5FmlIOJ/
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 https://web.whatsapp.com/ on Safari or Edge (before it became chromium based)
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 works by the alert message triggered by the click
21. ATTACK IS SUCCESSFUL

dmFyIHBheWxvYWQgPSAnYWxlcnQoNDU0NSk7JzsKcGF5bG9hZCA9ICdqYXZhc2NyaXB
0OiJodHRwczovL3d3dy5pbnN0YWdyYW0uY29tL3AvQjZVSDVGbWxsT0ovIjtldmFsKGF0b2
loIicgKyBidG9hKHBheWxvYWQpICsgJyIpKSc7CmUuX194X21hdGNoZWRUZXh0ID0gcGF
5bG9hZDsKZS5fX3hfYm9keSA9ICdJIGxvdmUgY2F0cyB3aXRoIGFsbCBtZSBoZWFydCEg
Y2hlY2tvdXQgdGhpcyBzdXBlciBjb29sIGNhdCBvbiBpbnN0YWdyYW0hICcgKyBwYXlsb2Fk
Ow==