Coinhive Proof of Work Widget Implementation

William Zhang

October 17, 2018

What is a CAPTCHA?

CAPTCHA is short for Completely Automated Public Turing Test to Tell Computers and Humans Apart. It is a program that protects websites against bots by creating and grading tests that only humans can pass but current computer program bots cannot. An example of a CAPTCHA is displayed in figure 1. Humans can read the distorted text in the CAPTCHA, but current computer programs cannot.

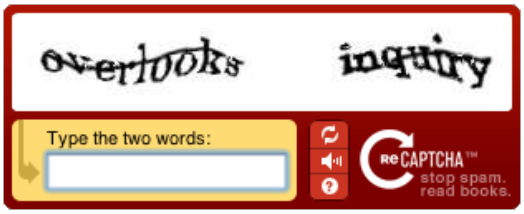

Figure 1: An example CAPTCHA

The Coinhive CAPTCHA

The Coinhive captcha works just like a conventional captcha. The proof of work is embedded in an HTML form, executes on the client side in the user's browser and generates a token. The token is then submitted together with the other data from the form. The validity of the toke is then checked on your server through the HTTP API that Coinhive provides on its website (https://coinhive.com/documentation/http-api).

In contrast with a conventional captcha however, the user does do the usual *"proof of human"*. Instead, the Coinhive captcha is a *"proof of work,"* – disincentivizing spammers to game your system as it is very uneconomic to do so.

Figure 2: Coinhive CAPTCHA at Work

Embedding the CAPTCHA

To embed the Coinhive captcha onto your website, you have to load the `captcha.min.js` anywhere on your page. `captcha.min.js` is loaded from a different domain (authedmine.com) to avoid adblockers. Then, create a `<div>` with the `coinhive-captcha` class for where you want the captcha to be displayed on your website. See figure 3 below for the implementation.

The text in the `<div>` below (*Loading Captcha...*) will be replaced by the captcha itself once it's loaded.

```
<form action="?" method="post">
    <!-- other form fields -->

    <script src="https://authedmine.com/lib/captcha.min.js" async></script>
    <div class="coinhive-captcha" data-hashes="1024" data-key="SITE_KEY">
        <em>Loading Captcha...<br>
        If it doesn't load, please disable Adblock!</em>
    </div>

    <input type="submit" value="Submit"/>
</form>
```

Figure 3: Embedding the Coinhive Captcha

Once the proof of work is completed, a field that is not displayed in figure 3 with the name `coinhive-captcha-token` will hold token name that was just generated by the captcha. Then this field is submitted with the rest of the form.

On the server side, you validate the submitted token through the Coinhive HTTP API using the command /token/verify.

```
curl -X POST \
    -d "token=<coinhive-captcha-token>" \
    -d "hashes=1024" \
    -d "secret=<secret-key>" \
    "https://api.coinhive.com/token/verify"

# {"success": true, "created": 1504205981, "hashes": 1024}
```

Figure 4: Verifying Token Validity

You want to specify the number of hashes twice. The first time on the client side for the widget, so it knows when it's done, and the second time when validating the submitted token on the server side, so the client can't trick the system.

Below is some information about what certain fields represent.

`data-key`	The public Site-Key
`data-hashes`	The number of hashes that the mining pool has to accept. The Coinhive pool uses a difficulty of 256. This means that your hash should be a multiple of 256.
`data-whitelabel`	Optional. Whether to conceal the Coinhive logo and the "What is this" link.
`data-callback`	Optional. The name of a global JavaScript function that should be called when the goal is reached.
`data-disable-elements`	Optional. A CSS selector for elements are disabled until the proof of work is complete. Usually in the form of a submit button.

Figure 5: Fields and Their What they Represent

Coinhive provides a full example of how to implement the captcha below:

```
<form action="?" method="post">
    <!-- other form fields -->
    <script src="https://authedmine.com/lib/captcha.min.js" async></script>
    <script>
        function myCaptchaCallback(token) {
            alert('Hashes reached. Token is: ' + token);
        }
    </script>
    <div class="coinhive-captcha"
        data-hashes="1024"
        data-key="SITE_KEY"
        data-whitelabel="false"
        data-disable-elements="input[type=submit]"
        data-callback="myCaptchaCallback"
    >
        <em>Loading Captcha...<br>
        If it doesn't load, please disable Adblock!</em>
    </div>

    <!-- submit button will be automatically disabled and later enabled
        again when the captcha is solved -->
    <input type="submit" value="Submit"/>
</form>
```

Figure 6: Example of how to Implement the Coinhive Captcha

Coinhive also provides an example of how to verify the submitted token using PHP on the server side.

```
$post_data = [
    'secret' => "SECRET-KEY", // <- Your secret key
    'token' => $_POST['coinhive-captcha-token'],
    'hashes' => 1024
];

$post_context = stream_context_create([
    'http' => [
        'header'  => "Content-type: application/x-www-form-urlencoded\r\n",
        'method'  => 'POST',
        'content' => http_build_query($post_data)
    ]
]);

$url = 'https://api.coinhive.com/token/verify';
$response = json_decode(file_get_contents($url, false, $post_context));

if ($response && $response->success) {
    // All good. Token verified!
}
```

Figure 7: Example Verify Token Using PHP

References

[1] Aleksandersen, Daniel. “On Using Coinhive CAPTCHA Instead of ReCAPTCHA by Google.” *Ctrl Blog*, CtrlBlog, 19 Nov. 2018, www.ctrl.blog/entry/coinhive-captcha.

[2] “Captcha.” *Terms of Service – Coinhive – Monero Mining Club*, coinhive.com/documentation/captcha.

[3] Hay, Steven. “Web Mining - Monetize Your Website through User Browsers.” *99 Bitcoins*, 99 Bitcoins, 22 Feb. 2018, 99bitcoins.com/webmining-monetize-your-website-through-user-browsers/.