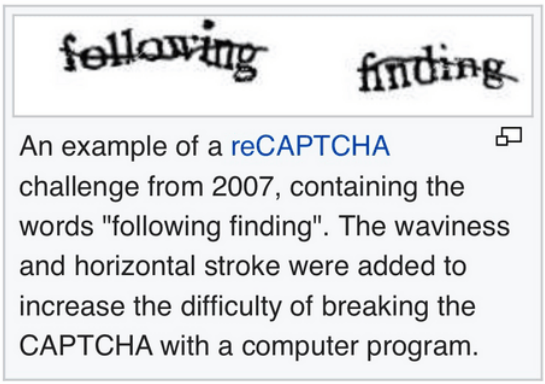
CAPTCHA Alternative

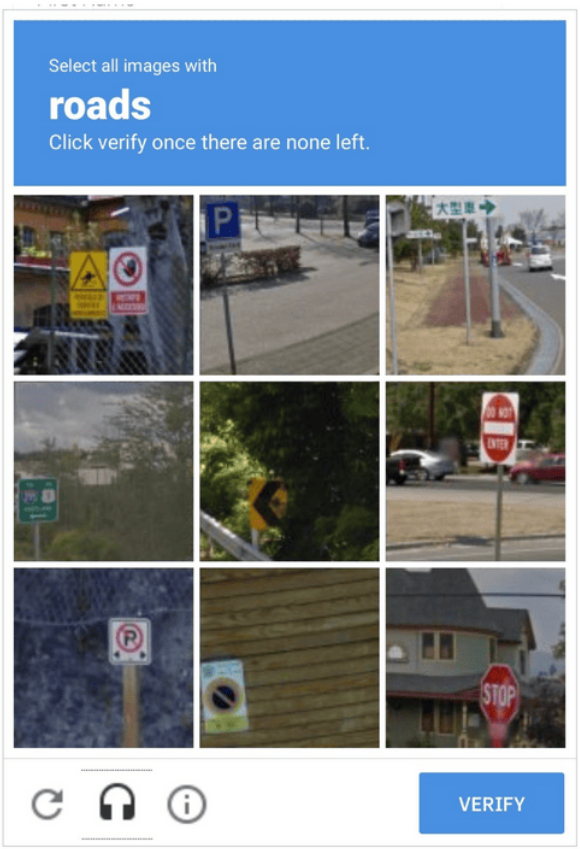
Protect your forms in a user friendly way

Willem L. Middelkoop
July 30, 2022



An example of a reCAPTCHA challenge from 2007, containing the words "following finding". The waviness and horizontal stroke were added to increase the difficulty of breaking the CAPTCHA with a computer program.

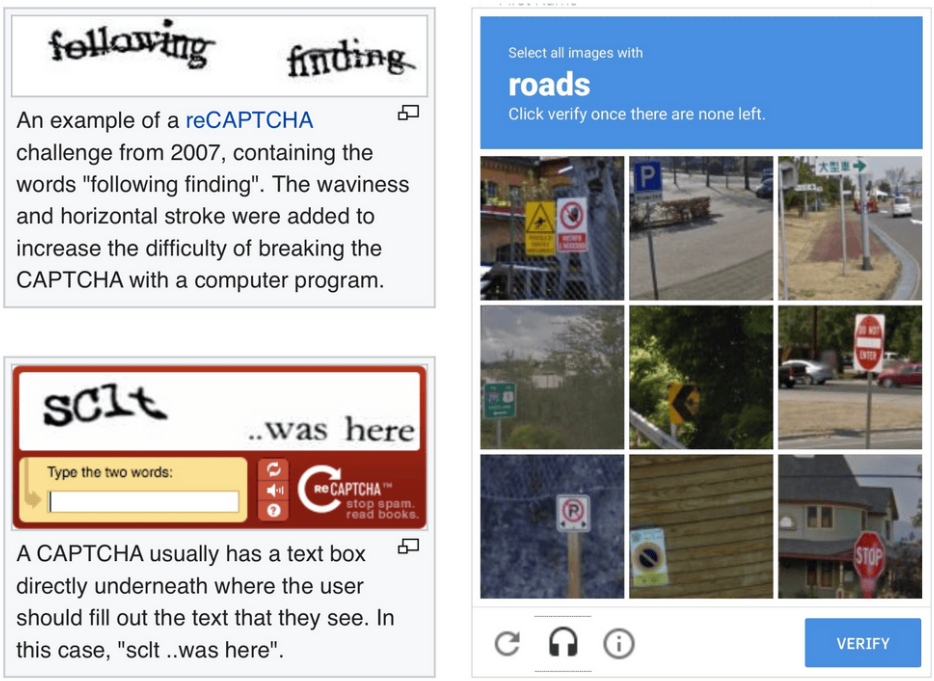
A CAPTCHA usually has a text box directly underneath where the user should fill out the text that they see. In this case, "sclt ..was here".



Chances are that you've encountered a CAPTCHA in the wild: they protect web forms by asking you to type over weirdly rendered characters or by asking you to select photos containing a particular thing. Why are they used and is there a user friendly alternative?

What is CAPTCHA?

Designed to tell computers and humans apart, CAPTCHA's often require you to do something that computer's can't easily do. Hence the acronym stands for: Completely Automated Public Turing test to tell Computers and Humans Apart. Web builders use CAPTHA's to protect their web forms against automated spam and abuse.

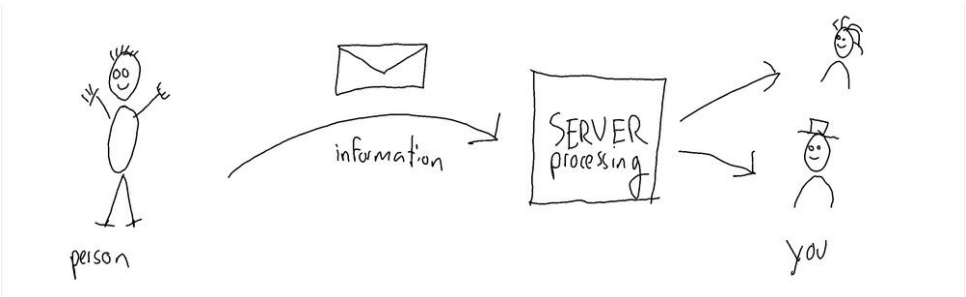


Typical CAPTCHA's you may encounter on the web (image credit: Wikipedia and Google)

Why do you need form protection?

A typical web form uses two steps:

- **enter information:** You enter some information and then you click "SUBMIT" (or something similar) to continue.
- **process information:** The information you entered is processed by a server, possibly forwarding your message or request to somebody's email.



A typical web form: server processes information from a person, forwarding it to other people, like yourself. Malicious actors can easily fake the sending person, enabling automated abuse of your processing server

The problem with this typical approach is that the first step can easily be automated by malicious actors. They use another computer to repeatedly submit information to the server, forcing it to process loads of information, often containing spam messages. Hackers are especially keen on finding web forms where they can manipulate the "TO"-address, because then they can use your server to send spam messages to anybody in the world.

A CAPTCHA protects your form by obstructing a computer from automatically submitting information to your server.

Problems with CAPTCHA

Frankly, I think they're ugly and cumbersome, and I am not alone. They introduce a threshold for people: lowering conversion rates, click through rates, etc. In addition, they are often very unfriendly to people with sub optimal sight.

Critics also complain about Google being one of the most popular suppliers of free CAPTCHA tests. One may wonder why they offer such a service free of charge? It is an excellent way for them to get in the loop of many (commercial) customer journeys, feeding their algorithms with more data to profile people. It is something to consider if you value privacy and GPDR/cookie laws.

Alternative

That's why I set out to create an effective alternative for CAPTCHA's to be used on the web forms that I build for my customers. My approach is two-fold:

- **hide:** most malicious actors search for possible targets by looking for any FORM-tag, which is often used to build web forms. My forms do not have the FORM-tag, but use other common elements such as DIV: creating a perfect camouflage. If they can't it, they can't break it.
- **harden:** my web forms use an additional step to make it harder to automate their abuse. I call it ”getting a post stamp” before submit.

Cryptographic pre-submit signature: the post stamp

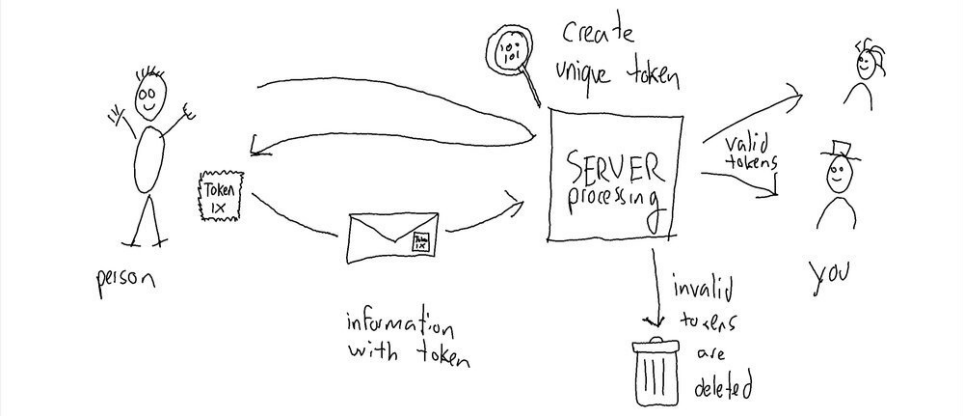
You must understand that malicious actors think economically: they will only do as much of an effort if the potential gain is worth it. If you make it hard enough for them to abuse your web form, chances are they'll move on and call it a day. Before I explain what I did, I must warn you:

Disclaimer: *You must consider the goal of your form protection, in my case it is (greatly) reducing the number of spam messages while preserving user-friendliness. This allows me to think of a system that does not necessarily provide ”bomb proof” protection, yet if you're trying to protect nuclear secrets then this approach might not fit you. Always consider your threat model!*

To protect web forms I introduce an extra step that I dubbed ”getting a post stamp”. The steps of the protected web form then become like this:

- **enter information:** You enter some information and then you click ”SUBMIT” (or something similar) to continue.
- **get post stamp:** Just before the actual information is submitted, your device contacts the server to ask for a cryptographic token that matches characteristics of the actual information *and* the sender. This happens in mere milliseconds, people won't notice it. Think of the token as a post stamp, specific for the package you're sending - to be used only once.

- **process information:** The information you entered is combined with the cryptographic token and posted to the server. If the token is valid, the information is processed as usual. Invalid tokens are simply ignored, which is important to make reverse engineering very hard by not providing any feedback (the server always says "OK" :-))!



By introducing a unique, use-only-once, token for each message it becomes much harder for malicious actors to automate the process of sending messages - protecting your form against spammers

Since 2016 I have been running this approach and the post processing servers have handled literally millions of messages, with extremely few messages (<100) being spam. Upon closer inspection, most of the spam that get through is "manual labour": possibly indicating malicious actors investigating the mechanism. As it appears to be too much of an effort to break the system, it is very effective in achieving my goal of protecting the web forms.

Conclusion

While CAPTCHA's are understandably very popular, it is not always necessary to have them on your web forms if you consider alternatives. It's my advice to make it harder for malicious actors and easier for your audience!