Is The CAPTCHA “DEAD”?

By
Peter Noblett

peterjanoblett@gmail.com

~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~

Index

Section 1

Section 2

1.1 INTRODUCTION

The Target Audience

This booklet is aimed at anyone involved in website design and development, including system architects and those actually commissioning a website to be built.

1.2 FIRST, SOME DEFINITIONS

Robots, Bots, Bad-Bots, DNS, Hackers and Spammers, CAPTCHA

We will be referring to robots (also know as “bots”), hackers and spammers. So here is a brief explanation of what they are:

A **bot** is short term for robot which is a computer program designed to automatically scan websites and read the pages and documents it finds. Bots are programmed to use the information they extract in a variety of ways depending on what the programmer has instructed it to do.

Good bots would be those that are collecting information for a genuine and honourable reason. Market research, updating of search engines. These are also know as web crawlers, they crawl the Internet collecting information as they go. Search engines use bots to build and maintain their vast databases.

For our purposes we are dealing with **bad-bots**, these are bots that are up to no good.

E.g.

- A simple type of bad-bot would be one looking for email addresses to add to a bulk list to sell to those who send out junk email (spammers).
- A more sophisticated one could be designed to extract data that may be used for identity theft.

DNS, Denial of Service attack, this is where someone swamps a website with so much traffic that legitimate users cannot use the site with any degree of reliability. In some cases the amount of traffic will cause the server to fail (crash).

A **hacker** is a physical person who will manually try and extract data from a website. They may start by using one or more bots to identify potentially vulnerable websites before launching a manual attack. They are often willing to spend a considerable amount of time and effort in achieving their goal. Hackers are driven by a variety of factors including financial reward (selling the information or using the information for theft or fraud), sheer malice or the fact that they regard it as a perverted game (intellectual challenge).

Hackers may not work alone. In parts of the world where labour is very cheap they may manage a group of hackers each one following the instructions of the “hack master”.

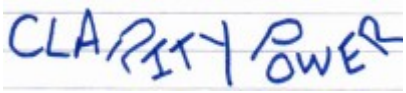
A **spammer** is someone who sends out spam emails. These emails often contain links to documents which contain code designed to extract sensitive data (account numbers, passwords etc.) stored on the user’s device and send it back to the spammer.

CAPTCHA

It is the acronym for:

Completely
Automated
Public
Turing tests to tell
Computers and
Humans
Apart

The commonest format uses letters or numbers displayed as a distorted image. The human brain scans this image and instantly recognises the letters. Whereas character recognition software has difficulty because not only are letters distorted (i.e. The first R and last R) but others overlap making it difficult to detect where one character ends and the next starts.



In simple terms a **CAPTCHA** is a visual element that, when displayed on the user's screen, can be easily identified by a human but not by a bot.

1.3 WHY DID WE START USING CAPTCHAs

The term CAPTCHA first appeared in 2003. However, the concept was detailed in a 1998 American patent therefore, in IT terms, the idea has been around for some time.

In fact it started in 1997. The first "CAPTCHA" was used to stop URLs being automatically submitted to a search engine called Alta Vista. In those days you had to pay to have your details included in a search engine. Also at that time we were using dial up modems with a download speed of 1,200 bits per second! Give that to a kid today and they would soon be back out on the street kicking a ball around.

Having just ordered a 50MB broadband line I have calculated that it represents an increase of 4,166,666% over my first modem which I had about 20 years ago. That works out at an average increase in speed of around 2,000% a year. I cannot think of any other technology that has changed at such a pace and over such a relatively short period. This increased speed has made it possible to both collect and disseminate data at a tremendous rate.

It also means that a single bot can now make thousands of website visits per second. Imagine if everyone in a football stadium tried to steal their neighbours wallet at the same time, there would be public uproar. Unfortunately that is what is happening on the Internet every minute of every day, 24 / 7. The majority of the general public have no idea what is really happening out there on the web. One reason is that web developers have used CAPTCHAs and other technology to help protect them.

What actually does a bad-bot do, it does whatever it has been programmed to do. Here are some more examples:

- Let assume we are using two bots, the first builds a list of 100,000 UK email addresses. A small % of those 100,000 will have used one of the commonest 1,000 passwords. We use a second bot to start trying every password for every email address and statistically we will get a few hits. Hey presto, we have access to a few accounts. We now repeat the process for another 100 or so sites and you now see we have got control of many more accounts. Bots are a numbers game, once set-up they can run for weeks on end.

- Denial of Service (DNS) attack. You have a simple enquiry form on your website. By setting a bad-bot to fill out the form 100,000 times your website is soon going to crash. It is also going to take a lot of effort to identify and clear out those unwanted enquiries.

- Organised criminals exchange details of bank cards. The details on the cards would be used to create dummy accounts. False but working email addresses are easy. Register a Gmail account e.g. iamacrook@gmail.com you can then use iamacrook+1@gmail.com, iamacrook+2@gmail.com etc. They then try and buy items using those cards to see which ones are still live.

1.4 RISKS

For the typical business website an attack by a robot is high whereas one by a hacker is relatively low. To be attacked by a hacker there has to be a good reason for them to expend their time and effort. A website selling small high value items items such as gold jewellery or branded perfumes might become a target for a hacker. However, there are other reasons why they may get attacked such as a disgruntled ex-employee or someone with a personal issue with the site owners or their ethics.

1.5 OTHER TYPES OF CAPTCHA

Although the letter / number version is the most common there are other ways of trying to fool those bad boys, these include:

Name	Method	Weakness
SimpleCaptcha	User solves a simple maths problem displayed as text	The problem has to be relatively simple as there are people who are "number blind". I have a friend who so is brilliant with words that she has worked as a proof reader. However she failed the maths test for a shelf stacking job in her local super store! As she says "I don't do numbers".
MathCaptcha	As SimpleCaptcha but the problem is displayed as an image.	Same as SimpleCaptcha.
QuestyCaptcha	Users have to answer a question from a series of questions	This has to be limited to sites where you know the IQ level of your users is high enough for them to have a very high probability of knowing the answer. You cannot presume an average IQ, because, by definition there are as many people below the average IQ as there are above. If the bad-bot / spammer can build a list of all of the questions then the door is open.

"Are you human"	Users have to answer a puzzle, often involving shapes and or colours.	By definition the puzzles have to be simple as not everyone can quickly solve them. A common one is a slide bar and the user is asked to move the pointer from one side to the other. Using colours could cause issues for those with colour blindness, 8% of men are colour blind whilst it only effects 1 in 200 women.

1.6 WHAT ARE THE ISSUES WITH USING A CAPTCHA

A CAPTCHA is not user friendly and some regard them as the second most irritating thing on the web after pop-ups. They can turn users off, if you are competing with hundreds of others out there on the Internet then you have to make the user experience on your website, if not enjoyable, then hassle free. It is like finding that someone has left a supermarket trolley in the middle of the last empty space in the parking area. The shop may be wonderful but you are going to walk through that door feeling annoyed with the lazy idiot who left the trolley in the way. The same with websites; a CAPTCHA can put people in the wrong frame of mind and one more little annoyance on your website and they will probably be off somewhere else.

"[CAPTCHA] present a major problem to users who are blind, have low vision or have a learning disability such as dyslexia", www.w3.org/TR/turingtest/

They can be a total nightmare on mobile devices. I recall being desperate to get hold of some technical info which required me registering on a site. That site was not particularly mobile friendly. I had to find my reading glasses then try numerous refreshes to get a CAPTCHA that I could actually read on my mobile phone.

And it is only getting worse, as the bots become clever at reading CAPTCHAs they become ever more obscure. It has now come to the stage that with certain types, especially those with murky house numbers, you might as well ignore the first one that is displayed; just hit the refresh button twice to get one that is easier to read. I bet some bad-bot writer out there has already worked that one out.

1.7 WEBSITES WITH FORMS

Large organisations will have quite sophisticated ways of protecting their users, e.g. Facebook monitors the geographical locations you log in from and may lock your account if it detects a login from an unexpected location.

The information in this guide is primarily for the benefit of those working on the millions of other websites that have to be built with more modest resources and budgets.

1.8 HOW DOES A BAD-BOT WORK

It is fairly simple to write a program that can read files from a website just like your browser does. The website will pass back to the bot the web page. Unlike your browser the bad-bot can then manipulate the contents of the page before sending it back to the website.

The bad-bot can look for input fields and then insert data into those fields, it could even remove fields or add additional ones. The bad-bot then posts [submits] the form back to the website, like a normal browser. The web server hosting that website processes the data as if it had been entered manually; it will usually reply by posting back another page to the bad-bot.

The bad-bot could be programmed to look for a response to see if the anticipated result has been achieved. The person coding the bad-bot will have a good idea what a successful response looks like and will be checking for it. If it thinks it has been successful it will proceed to the next task, page or website. If a satisfactory response is not received it may be programmed to try a variety of different data permutations before giving up and going to the next website.
It is therefore a key feature that any alternative to a CAPTCHA should block the attack and also return a valid message to fool the bad-bot into believing it has worked. This then poses the problem, how can we easily test for a valid update against a false update? One simple method is to manipulate the response in a subtle way such that those doing any QA testing or end user support can quickly recognize whether they are dealing with a valid or false response.

The bad-bot could be searching the response for some specific text i.e. “Your Application Has Been Approved”. Using different colours or enclosing one version of the message text in quotes or other symbols are simple methods that keep the text intact and should therefore still fool the bad-bot.

Another key point with bad-bots is that the programmer could have designed it to do several different things. So, as well as trying to complete data entry forms it could be collecting a lot of information from the pages it visits. So sending a response along the lines of “Got You—You Naughty Robot” will, to a hacker, be as a red rag is to a bull. They will probably attack your site just to prove they are clever than you. It is tempting to send them a rude message but always give them a response they think is normal.

Section 2

2.1 ALTERNATIVE METHODS

OVERVIEW

It is not expected that you copy our examples into your website but rather use the concept and snippets to create your own derivatives. Simply adding your own variations will make it harder for bad-bots. If we all used exactly the same code somebody out there would write a bad-bot to try and get through them all in one go.

For simplicity the code snippets are in HTML and classic ASP. Most other scripting languages have similar methods so you should be able to generate your own code from the concepts described here. Some of the methods could also be implemented using CSS.

You do not have to rely on a single method, incorporate a couple as this will make it more difficult for the bad boys out there; but do not make it over complex as it will become difficult to maintain.

2.2 Using Hidden Controls

The objective is to entice the robot to set a value to a control that a human user cannot access.

2.2.1 Fixed Hidden Controls

Pros: Simple to implement
Cons: Will fail if the bad-bot has been programmed to detect and ignore hidden fields

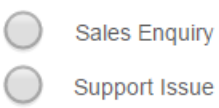
First, what controls would be the most effective?

- Check boxes: As we cannot predict the behaviour of the robot

we do not know whether it will ignore them, select one, select all, go into a loop and try different combinations? Because of the issue of this unpredictable behaviour we will ignore check boxes.

- Radio Buttons: If you have a set of radio buttons with the same name only one can be selected. You can ensure this by including the appropriate validation routine. Our assumption is the bad-bot is going to select either the first, last or all. If it selects all, the value will be taken from the last one of that set.

  If your web-page does not have radio buttons then it is fairly easy to add a simple question that enables you to add a set to your page. e.g. On a "Contact Us" form you could have:



  Add hidden controls before and after the visible ones; if you feel so inclined you could add one in the middle as well.

EXAMPLE – 2.2.1

```
<div style="display:none;">
    <input type="radio" name="myradio" id="rad1" value="1">First Hidden Enquiry
</div>
<div>
    <input type="radio" name="myradio" id="rad2" value="2">Sales Enquiry
</div>
<div>
    <input type="radio" name="myradio" id="rad3" value="3">Support Issue
</div>
<div style="display:none;">
    <input type="radio" name="myradio" id="rad4" value="4">Last Hidden Enquiry
</div>
```

In this example our validation code just needs to ensure that the value returned is 2 or 3 whilst the bad-bot is going to return 1 or 4.

Do not use labels such as First Hidden Field, Last Hidden Field etc. Use something that sounds plausible. In the example above one could use "Account Query", "Order Enquiry". Try not to give the game away!

2.2.2 Extra Input Field Hidden Behind Another Control

Although 2.2.1 is quick and simple to implement it does have the potential weakness whereby if a bad-bot has been designed to ignore hidden controls it may not work. A solution is to physically "hide" the control(s) when the form is loaded.

Pros: More robust, the bad-bot will not know an image or another control is hiding that control from the human user and that the human will ignore it.

Cons: Presumes the bad-bot enters data into every control.
Causes problems for visually impaired users.
Controls may be momentarily visible and therefore could cause issues with the user as to them wondering "what is happening".

In our example we get over this problem by:

1. Making the control small
2. Changing the colour so that it merges with the background
3. Obscure it by placing an image over the top

For simplicity, this example uses a text control, but you could uses the radio buttons from the example above, or other controls to make it more complex.

EXAMPLE – 2.2.2

```
<div id="yourdiv" >
    <input type="text" style="width:20px; color:#FFFFFF;" id="fred" name="fred" value="">
</div>
<img style="margin-top: -24px; width:140px;" src="your-small-image.jpg" >
```

You will need to adjust the margin-top value and so that it obscures the control(s) and the colour to match

your background.

In your validation check the value of field “fred” is still blank.

NB. You will need to consider catering for visually impaired users.

2.2.3 Adding An Extra Input Field Then Removing It On Form Load

Pros: Robust, the bad-bot would have to be able to read and process the JavaScript on the page. Due to the various ways you can use JavaScript this would make a very complex bad-bot. However this is the sort of security feature that a hacker would have a go at. Should not create issues for those with visual impairments.

Cons: Requires adding a JavaScript routine

EXAMPLE – 2.2.3

```
<div id="yourdiv" >
    <input type="text" style="width:20px; color:#FFFFFF;” id=”fred” name=”fred” value=”OK”>
</div>

//On load remove the div containing the control
<script type="text/javascript">
    (function () {
        var e = document.getElementById("yourdiv")
        e.parentNode.removeChild(e)
    })();
</script>
```

In your validation check the value of field “fred” is null.

2.3 Using Split Sites

This is a completely different approach; here we split the “marketing” activities and “back office” activities over two separate websites. It is not uncommon to have them on different websites for a variety of reasons including, where they are maintained by separate teams or there there are issues around:

- Accountability
- Maintenance
- Support
- Security

Another reason is to reduce the impact of a DNS attack. The “publicised” marketing website is probably the one most likely to be the target of an attack, by splitting them you reduce the possibility of loosing all services.

The objective here is to con the bad-bot into thinking it has worked when it has not.

We have to presume the bad-bot will visit either, or both sites, so we have to handle both scenarios.

The Marketing website will contain one or more links to the registration page.

i.e.

registerme.xxx

The first step is to move the registration page to the back office site. Secondly add a query-string pair (more about this later) to the URLs on the marketing site that calls the registration page:

i.e.

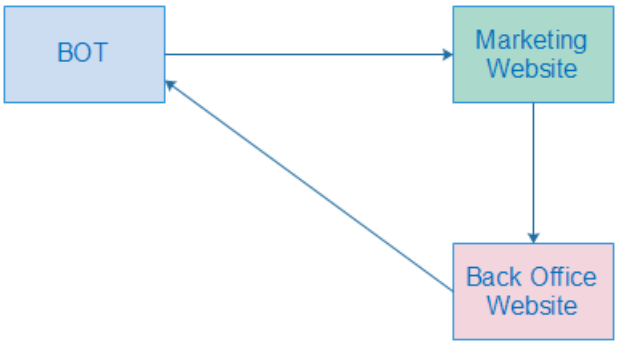
https:\\mybackoffice.com\registerme.xxx?qwerty=mnbvc

Notes.
1) Your back office site should always be running on a secure server, hence https in the example above. Beware, there are different types of secure server with different levels of security, the cheapest SSL certificate may not necessarily be the best for you. Please do some research before buying an SSL certificate.
2) The landing page on the back office site should be more obscure than resgisterme.xxx, more like 6GN78repy.xxx to make it harder for hackers; registerme.xxx is a bit of a give away. Your job is to make life difficult for bad-bots and hackers.

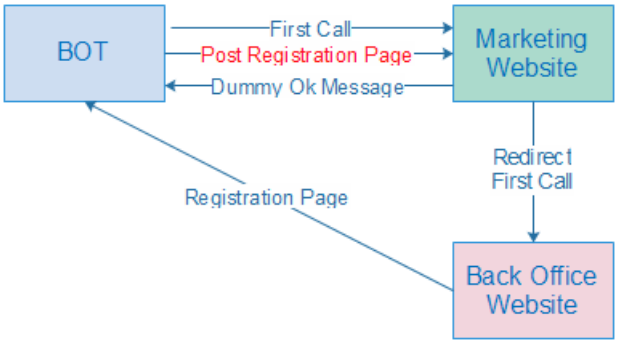
In reality it should be something like:

https:\\mybackoffice.com\6GN78repy.xxx?qwerty=mnbvc

The bad-bot calls the Marketing website which calls a page on the Back Office site which returns the registration page.



The bad-bot thinks the page has come from the Marketing Website, because that was the site it was processing, so it posts it back to the marketing website. You then have a page on the marketing website that has the same name as the registration page on the Back Office site and this page returns the expected message. e.g. "You Are Now Registered". The bad-bot will think it has been successful.



What happens if the bad-bot calls the registration page on the Back Office site?

This is where the query-string pair is required. You need to add some code to your registration page that tests the query-string pair is correct; if it is not correct then modify the registration page sent back.

What you send back is up to you, options include:

- Send back a dummy report. You have to remember that the it may be collecting information so you have to make it think it has hit a valid page.
- A simple message - “Thank you for visiting our site“ with a session abandon.
- If you have a page that is displayed when a user logs out then send that page if it seems appropriate.

Do not make it over complicated.

2.4 Using Behaviour

The bad-bot can fill and post back a form in a fraction of a second, a human takes a few seconds; we can use this information to catch it out. Measure how long it takes to correctly fill out your form. It is safe to say that if the time between sending the page and receiving it back is less than 80% of the minimum “human” time, it is either a bad-bot or someone quickly entering garbage, in either case we do not want to process that data.

The simplest way of testing for this is using a session variable. Store the date + time stamp when the page is created:

i.e.

```
Session(“yourtimestamp”) = Now
```

The page reading the data posted back compares the time stamp to the current time. If greater than the minimum you have allowed, process normally else treat as bad and respond appropriately.

i.e.

```
If DateDiff("s", Session(“yourtimestamp”) , Now) > 6 Then “human” Else “bot”
```

2.5 Conclusion

The three different techniques described here have all been successfully used on a variety of projects. On some sites two different methods were used on others more than one set of hidden fields were used.

Nothing in life is guaranteed and in no way are these suggested methods going to fully protect your site but hopefully I have given you some ideas on how you can avoid having to use a pesky CAPTCHA.

Peter Noblett
14 October 2015

Feedback:
E: peterjanoblett@gmail.com

Suggested Reading:
http://www.w3.org/TR/turingtest/

Acknowledgements:
https://en.wikipedia.org/wiki/CAPTCHA
https://www.mediawiki.org/wiki/CAPTCHA
https://www.mediawiki.org/wiki/Extention:ConfirmEdit