Stopping scripters from slamming your website

Asked 15 years, 11 months ago Modified 1 year, 11 months ago

Viewed 93k times

506

I've accepted an answer, but sadly, I believe we're stuck with our original worst case scenario: **CAPTCHA everyone on purchase attempts of the crap**. Short explanation: caching / web farms make it impossible to track hits, and any workaround (sending a non-cached web-beacon, writing to a unified table, etc.) slows the site down worse than the bots would. There is likely some pricey hardware from Cisco or the like that can help at a high level, but it's hard to justify the cost if CAPTCHA-ing everyone is an alternative. I'll attempt a more full explanation later, as well as cleaning this up for future searchers (though others are welcome to try, as it's community wiki).

Situation

This is about the bag o' crap sales on woot.com. I'm the president of Woot Workshop, the subsidiary of Woot that does the design, writes the product descriptions,

podcasts, blog posts, and moderates the forums. I work with CSS/HTML and am only barely familiar with other technologies. I work closely with the developers and have talked through all of the answers here (and many other ideas we've had).

Usability is a massive part of my job, and making the site exciting and fun is most of the rest of it. That's where the three goals below derive. CAPTCHA harms usability, and bots steal the fun and excitement out of our crap sales.

Bots are slamming our front page tens of times a second screen scraping (and/or scanning our RSS) for the Random Crap sale. The moment they see that, it triggers a second stage of the program that logs in, clicks I want One, fills out the form, and buys the crap.

Evaluation

> lc: On stackoverflow and other sites that use this method, they're almost always dealing with authenticated (logged in) users, because the task being attempted requires that.

On Woot, anonymous (non-logged) users can view our home page. In other words, the slamming bots can be non-authenticated (and essentially non-trackable except by IP address).

So we're back to scanning for IPs, which a) is fairly useless in this age of cloud networking and spambot zombies and b) catches too many innocents given the number of businesses that come from one IP address (not to mention the issues with non-static IP ISPs and potential performance hits to trying to track this).

Oh, and having people call us would be the worst possible scenario. Can we have them call you?

> BradC: Ned Batchelder's methods look pretty cool, but they're pretty firmly designed to defeat bots built for a network of sites. Our problem is bots are built specifically to defeat our site. Some of these methods could likely work for a short time until the scripters evolved their bots to ignore the honeypot, screen-scrape for nearby label names instead of form ids, and use a javascript-capable browser control.

> lc again: "Unless, of course, the hype is part of your marketing scheme." Yes, it definitely is. The surprise of when the item appears, as well as the excitement if you manage to get one is probably as much or more important than the crap you actually end up getting. Anything that eliminates first-come/first-serve is detrimental to the thrill of 'winning' the crap.

> novatrust: And I, for one, welcome our new bot overlords. We actually do offer RSSfeeds to allow 3rd party apps to scan our site for product info, but not ahead of the main site HTML. If I'm interpreting it right, your solution does help goal 2 (performance issues) by completely sacrificing goal 1, and just resigning the fact that bots will be buying most of the crap. I up-voted your response, because your last paragraph pessimism feels accurate to me. There seems to be no silver bullet here.

The rest of the responses generally rely on IP tracking, which, again, seems to both be useless (with botnets/zombies/cloud networking) and detrimental (catching many innocents who come from same-IP destinations).

Any other approaches / ideas? My developers keep saying "let's just do CAPTCHA" but I'm hoping there's less intrusive methods to all actual humans wanting some of our crap.

Original question

Say you're selling something cheap that has a very high perceived value, and you have a very limited amount. No one knows exactly when you will sell this item. And over a

million people regularly come by to see what you're selling.

You end up with scripters and bots attempting to programmatically [a] figure out when you're selling said item, and [b] make sure they're among the first to buy it. This sucks for two reasons:

1. Your site is slammed by non-humans, slowing everything down for everyone.
2. The scripters end up 'winning' the product, causing the regulars to feel cheated.

A seemingly obvious solution is to create some hoops for your users to jump through before placing their order, but there are at least three problems with this:

- The user experience sucks for humans, as they have to decipher CAPTCHA, pick out the cat, or solve a math problem.
- If the perceived benefit is high enough, and the crowd large enough, some group will find their way around any tweak, leading to an arms race. (This is especially true the simpler the tweak is; hidden 'comments' form, re-arranging the form elements, mis-labeling them, hidden 'gotcha' text all will work once and then need to be changed to fight targeting this specific form.)
- Even if the scripters can't 'solve' your tweak it doesn't prevent them from slamming your front page, and then sounding an alarm for the scripter to fill out the

order, manually. Given they get the advantage from solving [a], they will likely still win [b] since they'll be the first humans reaching the order page. Additionally, 1. still happens, causing server errors and a decreased performance for everyone.

Another solution is to watch for IPs hitting too often, block them from the firewall, or otherwise prevent them from ordering. This could solve 2. and prevent [b] but the performance hit from scanning for IPs is massive and would likely cause more problems like 1. than the scripters were causing on their own. Additionally, the possibility of cloud networking and spambot zombies makes IP checking fairly useless.

A third idea, forcing the order form to be loaded for some time (say, half a second) would potentially slow the progress of the speedy orders, but again, the scripters would still be the first people in, at any speed not detrimental to actual users.

Goals

1. Sell the item to non-scripting humans.
2. Keep the site running at a speed not slowed by bots.
3. Don't hassle the 'normal' users with any tasks to complete to prove they're human.

scripting **e-commerce** **bots** **detection**

Share

Improve this question

Follow

edited Jun 20, 2020 at 9:12

community wiki
14 revs, 6 users 53%
Dave Rutledge

1 I think you have contradicting goals: Keeping the experience exactly as it is but get rid of the bots. I think you can't get the one while not sacrificing a part of the other. – max Feb 7, 2009 at 8:57

It's a community wiki, so feel free to take a stab, but I was mostly trying to cover every point as clearly as I could considering there are obvious things to try that we'd already tried and discounted. – Dave Rutledge Feb 9, 2009 at 23:54

Why not just cache repeated offenders, simply don't update whatever page they're repeatably requesting. IPv4 and MAC addresses are 32 + 48 bits in total. That's 10MB for 1 million users, shouldn't be a problem. The combination IPv4 and MAC should help you track all kinds of users more accurately – John Leidegren Feb 13, 2009 at 7:42

5 I don't really understand why you need to let anonymous users see the crap sale. Why not only offer it to users who are logged in? If you do that, you wouldn't have unknown users hitting the page too often and then could ban bad users. – Ryan Guill Feb 13, 2009 at 14:48

2 I think some people are missing a key factor here: these bots are set up to log in and purchase too. They DO know a valid account and CAN be logged in. Also, real people that use woot sit there the minute an item is going to come up and hit

F5 to reload every 2-5 sec. That is valid normal human use.
– CodingWithSpike Feb 13, 2009 at 15:58

129 Answers

Sorted by: Highest score (default)

246

How about implementing something like SO does with the CAPTCHAs?

If you're using the site normally, you'll probably never see one. If you happen to reload the same page too often, post successive comments too quickly, or something else that triggers an alarm, make them prove they're human. In your case, this would probably be constant reloads of the same page, following every link on a page quickly, or filling in an order form too fast to be human.

+200

If they fail the check x times in a row (say, 2 or 3), give that IP a timeout or other such measure. Then at the end of the timeout, dump them back to the check again.

Since you have unregistered users accessing the site, you do have only IPs to go on. You can issue sessions to each browser and track that way if you wish. And, of course, throw up a human-check if too many sessions are being (re-)created in succession (in case a bot keeps deleting the cookie).

As far as catching too many innocents, you can put up a disclaimer on the human-check page: "This page may also appear if too many anonymous users are viewing our site from the same location. We encourage you to register or login to avoid this." (Adjust the wording appropriately.)

Besides, what are the odds that X people are loading the same page(s) at the same time from one IP? If they're high, maybe you need a different trigger mechanism for your bot alarm.

Edit: Another option is if they fail too many times, and you're confident about the product's demand, to block them and make them personally CALL you to remove the block.

Having people call does seem like an asinine measure, but it *makes sure there's a human somewhere behind the computer*. The key is to have the block only be in place for a condition which should almost never happen unless it's a bot (e.g. fail the check multiple times in a row). Then it FORCES human interaction - to pick up the phone.

In response to the comment of having them call me, there's obviously that tradeoff here. Are you worried enough about ensuring your users are human to accept a couple phone calls when they go on sale? If I were so concerned about a product getting to human users, I'd have to make this decision, perhaps sacrificing a (small) bit of my time in the process.

Since it seems like you're determined to not let bots get the upper hand/slam your site, I believe the phone may be a good option. Since I don't make a profit off your product, I have no interest in receiving these calls. Were you to share some of that profit, however, I may become interested. As this is your product, you have to decide how much you care and implement accordingly.

The other ways of releasing the block just aren't as effective: a timeout (but they'd get to slam your site again after, rinse-repeat), a long timeout (if it was really a human trying to buy your product, they'd be SOL and punished for failing the check), email (easily done by bots), fax (same), or snail mail (takes too long).

You could, of course, instead have the timeout period increase per IP for each time they get a timeout. Just make sure you're not punishing true humans inadvertently.

Share Improve this answer Follow

edited Feb 9, 2009 at 0:50

community wiki
6 revs
lc.

---

13 Google uses this same approach, and they only have IP addresses to go on. Frequently at work I'll get a CAPTCHA before I can search on Google because they see bot-like behavior from the same IP address. I think this approach

(CAPTCHA after bot-like behavior) is the best you're going to get. – Ross Feb 7, 2009 at 17:01

8 I've had google ask me for a CAPTCHA before, but it was my own fault - I was using them as a calculator, doing dozens of nearly-identical sums. – Marcus Downing Feb 7, 2009 at 17:54

The CAPTCHA option sounds like a winner to me. You hurt the bots hard and if well balanced you should never get in your legitimate users way. – xan Feb 12, 2009 at 19:07

Instead of locking people out and using a phone call, could you generate a temporary email address like cur92Siva@site.com, but generate the front part with an image. – Sam Feb 13, 2009 at 3:00

That might work too, unless the bots just get used to the system and can screen-scrape the email address. My point with the phone call is it actually forces human interaction and requires the user to explain themselves directly with their voice. Bot owners probably don't want to do that. – lc. Feb 13, 2009 at 4:17

205

You need to figure a way to make the bots buy stuff that is massively overpriced: 12mm wingnut: $20. See how many bots snap up before the script-writers decide you're gaming them.

Use the profits to buy more servers and pay for bandwidth.

Share Improve this answer edited Feb 13, 2020 at 3:22

Follow

community wiki
2 revs, 2 users 89%
Christopher Mahan

16 What if they then return the items or issue a chargeback? This could end up costing you and chargebacks can hurt your business with credit card processors. The bots are also likely using stolen cards, but that may exacerbate the level of chargebacks as higher amounts will be challenged more often. – Tai Squared Feb 9, 2009 at 18:53

17 Don't charge them, but mark them as bots, specifically for trying to buy the item. If any body buys a phoney item, then just mark them as a bot, and disallow them. You could probably just lock them out for a few hours. – Kibbee Feb 10, 2009 at 1:45

5 This has serious comedy value, until you anger a script-kiddie that happens to have more skills than just scraping woot, and causes you real problems because you ripped him off. – MattBelanger Feb 13, 2009 at 3:50

2 If the script kiddie gets angry they might just expose themselves enough for you to tag them and hand them over to law-enforcement. – Jacco Feb 13, 2009 at 12:21

11 sqook: this is not a technology solution, but a real world solution. Putting security guards with guns in banks is the same thing. It may seem hard-nosed, but so are the crooks, so be hard-nosed. Hurt them where it hurts until they stop. – Christopher Mahan Feb 14, 2009 at 15:36

167

My solution would be to make screen-scraping worthless by putting in a roughly 10 minute delay for 'bots and scripts.



Here's how I'd do it:

- **Log and identify any repeat hitters.**

You don't need to log every IP address on every hit. Only track one out of every 20 hits or so. A repeat offender will still show up in a randomized occassional tracking.

- **Keep a cache of your page from about 10-minutes earlier.**
- **When a repeat-hitter/bot hits your site, give them the 10-minute old cached page.**

They won't immediately know they're getting an old site. They'll be able to scrape it, and everything, but they won't win any races anymore, because "real people" will have a 10 minute head-start.

Benefits:

- No hassle or problems for users (like CAPTCHAs).
- Implemented fully on server-side. (no reliance on Javascript/Flash)
- Serving up an older, cached page should be less performance intensive than a live page. You may actually decrease the load on your servers this way!

Drawbacks

- Requires tracking some IP addresses

- Requires keeping and maintaining a cache of older pages.

Share Improve this answer Follow

edited Sep 8, 2022 at 8:10

community wiki
6 revs, 2 users 93%
abelenky

1 Damn it. I just spent an hour and a half writing up my own five-vector scheme for woot, and after thinking long and hard over my fifth countermeasure (a botnet throttle), I had to admit defeat. It doesn't work. And the rest of my hour-long solution is -- well, this one. abelenky, I tip my hat to you – Jens Roland Feb 8, 2009 at 20:09

7 To build on top of this: Put the IPs into an in-memory LRU counting hash (increment and push to top every time an IP comes back). Add heuristics based on reverse IP info, activity, image/js/cookie downloads. Scale your response by how bad the attack is, minimizing consequences of false negatives. – SquareCog Feb 11, 2009 at 13:30

1 (continued:) And my technique doesn't shut-out / ban anyone. It just gives them delayed information. No one in the office may win a prize, but that isn't much a problem from a customer-service / accessibility viewpoint. – abelenky Feb 13, 2009 at 0:03

20 @bruceatk: If you give them a special bots-only page, they will eventually learn to detect it, and learn to spoof a regular client more accurately. By giving old page, they will have NO IDEA that they are receiving old data. The old data is legitimate! Its just useless for contest/race purposes. – abelenky Feb 13, 2009 at 3:57

1 Big thanks to those who upvoted my idea. Even though the bounty is over, I think this idea has lots of merit in terms of being easier to implement than a captcha, less likely to harass humans, and more likely to foil bots. I hope someone gives this a try on some website. – abelenky Feb 13, 2009 at 19:51

56

Take a look at this article by ned Batchelder here. His article is about stopping spambots, but the same techniques could easily apply to your site.

> Rather than stopping bots by having people identify themselves, we can stop the bots by making it difficult for them to make a successful post, or by having them inadvertently identify themselves as bots. This removes the burden from people, and leaves the comment form free of visible anti-spam measures.
>
> This technique is how I prevent spambots on this site. It works. The method described here doesn't look at the content at all.

Some other ideas:

- Create an *official* auto-notify mechanism (RSS feed? Twitter?) that people can subscribe to when your product goes on sale. This reduces the *need* for people to make scripts.

- Change your obfuscation technique *right before* a new item goes on sale. So even if the scripters can escalate the arms race, they are always a day behind.

EDIT: To be totally clear, Ned's article above describe methods to prevent the automated PURCHASE of items by preventing a BOT from going through the forms to submit an order. His techniques wouldn't be useful for preventing bots from screen-scraping the home page to determine when a Bandoleer of Carrots comes up for sale. I'm not sure preventing THAT is really possible.

With regard to your comments about the effectiveness of Ned's strategies: Yes, he discusses honeypots, but I don't think that's his strongest strategy. His discussion of the **SPINNER** is the original reason I mentioned his article. Sorry I didn't make that clearer in my original post:

> The spinner is a hidden field used for a few things: it hashes together a number of values that prevent tampering and replays, and is used to obscure field names. The spinner is an MD5 hash of:
>
> - The timestamp,
> - The client's IP address,
> - The entry id of the blog entry being commented on, and

> - A secret.

Here is how you could implement that at WOOT.com:

Change the "secret" value that is used as part of the hash each time a new item goes on sale. This means that if someone is going to design a BOT to auto-purchase items, *it would only work until the next item comes on sale*!!

Even if someone is able to quickly re-build their bot, all the other actual users will have already purchased a BOC, and your problem is solved!

The other strategy he discusses is to *change* the honeypot technique from time to time (again, change it when a new item goes on sale):

- Use CSS classes (randomized of course) to set the fields or a containing element to display:none.
- Color the fields the same (or very similar to) the background of the page.
- Use positioning to move a field off of the visible area of the page.
- Make an element too small to show the contained honeypot field.
- Leave the fields visible, but use positioning to cover them with an obscuring element.
- Use Javascript to effect any of these changes, requiring a bot to have a full Javascript engine.

- Leave the honeypots displayed like the other fields, but tell people not to enter anything into them.

I guess my overall idea is to CHANGE THE FORM DESIGN when each new item goes on sale. Or at LEAST, change it when a new BOC goes on sale.

Which is what, a couple times/month?

Share Improve this answer Follow

edited Sep 8, 2022 at 8:10

community wiki
3 revs, 2 users 98%
BradC

+1 for the RSS. Make it so that legitimate users are rewarded. – Marcus Downing Feb 7, 2009 at 18:07

RSS seems like a good solution, but might that hurt the ad revenue that I am guessing this site depends on? – TM. Feb 7, 2009 at 20:53

2 I don't quite understand the "spinner" concept. Is this just an extra piece of data that is placed inside an html `<form>` and sent upon submission? Because a bot can easily scrape that as well. – Ponkadoodle Sep 6, 2013 at 19:22

45

Q: How would you stop scripters from slamming your site hundreds of times a second?
A: You don't. There is no way to *prevent* this behavior by external agents.

You could employ a vast array of technology to analyze incoming requests and heuristically attempt to determine who is and isn't human...but it would fail. Eventually, if not immediately.

The only viable long-term solution is to *change the game* so that the site is not bot-friendly, or is less attractive to scripters.

How do you do that? Well, that's a different question! ;-)

...

OK, some options have been given (and rejected) above. I am not intimately familiar with your site, having looked at it only once, but since people can read text in images and bots cannot easily do this, change the announcement to be an image. *Not a CAPTCHA*, just an image -

- generate the image (cached of course) when the page is requested
- keep the image source name the same, so that doesn't give the game away
- most of the time the image will have ordinary text in it, and be aligned to appear to be part of the inline HTML page
- when the game is 'on', the image changes to the announcement text
- the announcement text reveals a url and/or code *that must be manually entered* to acquire the prize.

CAPTCHA the code if you like, but that's probably not necessary.

- for additional security, the code can be a one-time token generated specifically for the request/IP/agent, so that repeated requests generate different codes. Or you can pre-generate a bunch of random codes (a one-time pad) if on-demand generation is too taxing.

Run time-trials of real people responding to this, and ignore ('oops, an error occurred, sorry! please try again') responses faster than (say) half of this time. This event should also trigger an alert to the developers that at least one bot has figured out the code/game, so it's time to change the code/game.

Continue to change the game periodically anyway, even if no bots trigger it, just to waste the scripters' time. Eventually the scripters should tire of the game and go elsewhere...we hope ;-)

One final suggestion: when a request for your main page comes in, *put it in a queue* and respond to the requests in order in a separate process (you may have to hack/extend the web server to do this, but it will likely be worthwhile). If another request from the same IP/agent comes in while the first request is in the queue, ignore it. This should automatically shed the load from the bots.

EDIT: another option, aside from use of images, is to use javascript to fill in the buy/no-buy text; bots rarely interpret

javascript, so they wouldn't see it

Share Improve this answer Follow

edited Feb 13, 2009 at 5:29

community wiki
4 revs
Steven A. Lowe

1 I would make sure that the "default text" changes also. This would prevent the scraping app from just comparing the image to a previous image and waiting for a significant change. +1. Great idea. – Frank Krueger Feb 7, 2009 at 9:10

1 Amendment to the "final suggestion": If a second request comes in from an address while a previous request from the same address is pending, discard the first request and put the second one in the queue. This will act as a penalty for hammering the site instead of letting the page load. – Dave Sherohman Feb 7, 2009 at 18:12

@[Frank Krueger]: i thought i implied this, but upon re-reading i guess i didn't - thanks for pointing it out! It might also be useful to have the default-text image change just a few pixels to mess with comparisons, and/or generate nearly invisible watermark-style text to further mess with bots – Steven A. Lowe Feb 12, 2009 at 3:04

@[Dave Sherohman]: you could, but that might cause the queue to churn; it may be better to just discard the new requests to shed the load immediately - testing/profiling would tell for certain which is better, but thanks for a good suggestion! – Steven A. Lowe Feb 12, 2009 at 3:06

Can't stand it that you told him to basically give in, I know you think its impossible, but I disagree. If there is a will, there is always certainly a way. Allowing defeat so easily is really

uninspiring and saddening, if the orig poster is reading, it is possible to do, but the solution will need to be custom designed after analysis of traffic logs, you can prevent current methods and future proof it to prevent yet unused methods. Also re JavaScript, the webbrowser control runs JavaScript in real time, no need for another engine - they can mess with the Dom and run their own JavaScript! Opps
– Erx_VB.NExT.Coder Sep 2, 2012 at 17:46

31

I don't know how feasible this is: ... go on the offensive.

Figure out what data the bots are scanning for. Feed them the data that they're looking for when you're NOT selling the crap. Do this in a way that won't bother or confuse human users. When the bots trigger phase two, they'll log in and fill out the form to buy $100 roombas instead of BOC. Of course, this assumes that the bots are not particularly robust.

Another idea is to implement random price drops over the course of the bag o crap sale period. Who would buy a random bag o crap for $150 when you CLEARLY STATE that it's only worth $20? Nobody but overzealous bots. But then 9 minutes later it's $35 dollars ... then 17 minutes later it's $9. Or whatever.

Sure, the zombie kings would be able to react. The point is to make their mistakes become very costly for them (and to make them pay you to fight them).

All of this assumes you want to piss off some bot lords, which may not be 100% advisable.

Share Improve this answer Follow

edited Feb 8, 2009 at 2:01

community wiki
2 revs, 2 users 94%
Zac Thompson

Don't think pissing off bot lords is desirable, but you have an interesting idea here. – Shawn Miller Feb 7, 2009 at 17:02

7 I agree, and I'm liking this repeating idea of fooling the bots into making bogus purchases. It's payback, and since they're breaking the ToS already, they can hardly complain. – Blank Feb 11, 2009 at 18:32

22

So the problem really seems to be: the bots want their "bag 'o crap" because it has a high perceived value at a low perceived price. You sometimes offer this item and the bots lurk, waiting to see if it's available and then they buy the item.

Since it seems like the bot owners are making a profit (or potentially making a profit), the trick is to make this unprofitable for them by *encouraging* them to buy the crap.

First, *always* offer the "bag 'o crap".

Second, make sure that crap is usually crap.

Third, rotate the crap frequently.

Simple, no?

You'll need a permanent "why is our crap sometimes crap?" link next to the offer to explain to humans what's going on.

When the bot sees that there's crap and the crap is automatically purchased, the recipient is going to be awfully upset that they've paid $10 for a broken toothpick. And then an empty trash bag. And then some dirt from the bottom of your shoe.

If they buy enough of this crap in a relatively short period of time (and you have large disclaimers all over the place explaining why you're doing this), they're going to lose a fair "bag 'o cash" on your "bag 'o crap". Even human intervention on their part (checking to ensure that the crap isn't crap) can fail if you rotate the crap often enough. Heck, maybe the bots will notice and not buy anything that's been in the rotation for too short a time, but that means the humans will buy the non-crap.

Heck, your regular customers might be so amused that you can turn this into a huge marketing win. Start posting how much of the "crap" carp is being sold. People will come back just to see how hard the bots have been bitten.

Update: I expect that you might get a few calls up front with people complaining. I don't think you can stop that entirely. However, if this kills the bots, you can always stop it and restart it later.

Share Improve this answer Follow

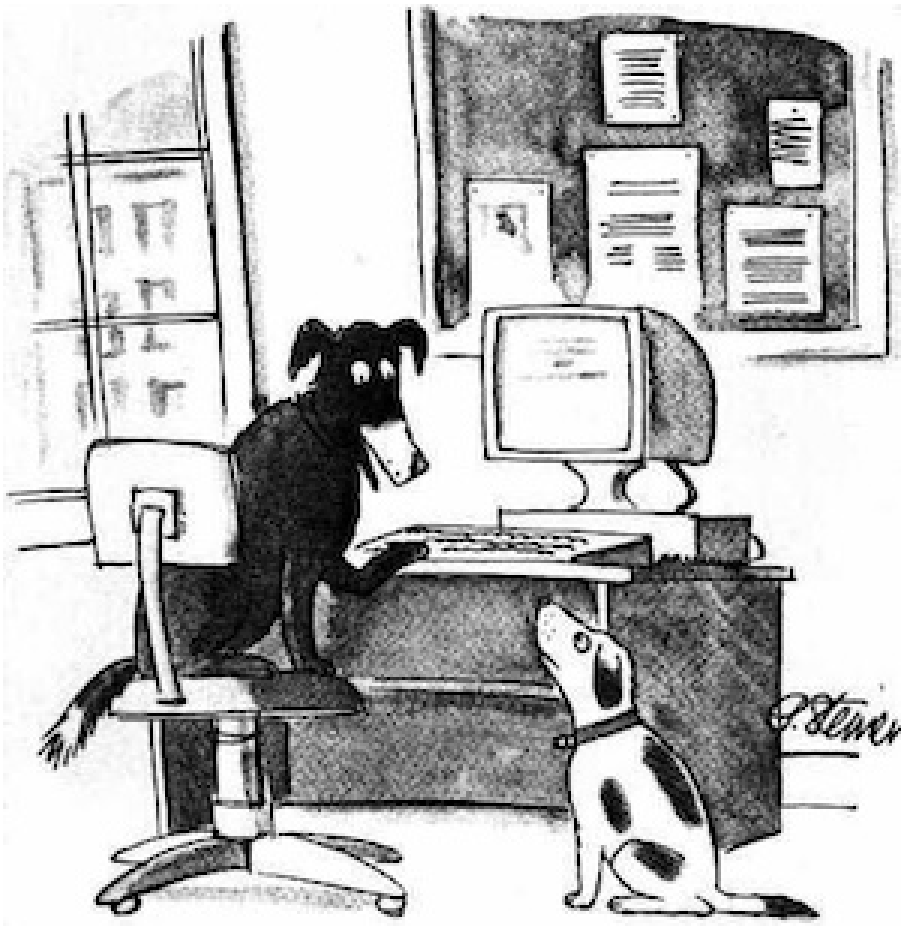
answered Feb 13, 2009 at 8:36

community wiki
Ovid

18

1. Sell the item to non-scripting humans.
2. Keep the site running at a speed not slowed by bots.
3. Don't hassle the 'normal' users with any tasks to complete to prove they're human.

You probably don't want to hear this, but #1 and #3 are mutually exclusive.



"On the Internet, nobody knows you're a dog."

Well, nobody knows you're a bot either. There's no programatic way to tell the whether or not there's a human on the other end of the connection without requiring the person to do something. Preventing scripts/bots from doing stuff on the web is the whole reason CAPTCHAs were invented. It's not like this is some new problem that hasn't seen a lot of effort expended on it. If there were a better way to do it, one that didn't involve the hassle to real users that a CAPTCHA does, everyone would be using it already.

I think you need to face the fact that if you want to keep bots off your ordering page, a good CAPTCHA is the only way to do it. If demand for your random crap is high enough that people are willing to go to these lengths to

get it, legitimate users aren't going to be put off by a CAPTCHA.

Share Improve this answer Follow

edited Feb 8, 2017 at 14:10

community wiki
2 revs
Chris Upchurch

+1 for if they want it, a captcha ain't going to stop them ... and for the cartoon. – Martin Oct 10, 2009 at 20:23

14

The method Woot uses to combat this issue is changing the game - literally. When they present an extraordinarily desirable item for sale, they make users play a video game in order to order it.

Not only does that successfully combat bots (they can easily make minor changes to the game to avoid automatic players, or even provide a new game for each sale) but it also gives the impression to users of "winning" the desired item while slowing down the ordering process.

It still sells out very quickly, but I think that the solution is good - re-evaluating the problem and changing the parameters led to a successful strategy where strictly technical solutions simply didn't exist.

Your entire business model is based on "first come, first served." You can't do what the radio stations did (they no longer make the first caller the winner, they make the 5th or 20th or 13th caller the winner) - it doesn't match your primary feature.

No, there is no way to do this without changing the ordering experience for the real users.

Let's say you implement all these tactics. If I decide that this is important, I'll simply get 100 people to work with me, we'll build software to work on our 100 separate computers, and hit your site 20 times a second (5 seconds between accesses for each user/cookie/account/IP address).

You have two stages:

1. Watching front page
2. Ordering

You can't put a captcha blocking #1 - that's going to lose real customers ("What? I have to solve a captcha each time I want to see the latest woot?!?").

So my little group watches, timed together so we get about 20 checks per second, and whoever sees the change first alerts all the others (automatically), who will load the front page once again, follow the order link, and perform the transaction (which may also happen automatically, unless you implement captcha and change it for every wootoff/boc).

You can put a captcha in front of #2, and while you're loathe to do it, that may be the only way to make sure that even if bots watch the front page, real users are getting the products.

But even with captcha my little band of 100 would still have a significant first mover advantage - and there's no way you can tell that we aren't humans. If you start timing our accesses, we'd just add some jitter. We could randomly select which computer was to refresh so the order of accesses changes constantly - but still looks enough like a human.

First, get rid of the simple bots

You need to have an adaptive firewall that will watch requests and if someone is doing the obvious stupid thing - refreshing more than once a second at the same IP then employ tactics to slow them down (drop packets, send back refused or 500 errors, etc).

This should significantly drop your traffic and alter the tactics the bot users employ.

Second, make the server blazingly fast.

You really don't want to hear this... but...

I think what you need is a fully custom solution from the bottom up.

You don't need to mess with TCP/IP stack, but you may need to develop a very, very, very fast custom server that is purpose built to correlate user connections and react appropriately to various attacks.

Apache, lighthttpd, etc are all great for being flexible, but you run a single purpose website, and you really need to be able to both do more than the current servers are capable of doing (both in handling traffic, and in appropriately combating bots).

By serving a largely static webpage (updates every 30 seconds or so) on a custom server you should not only be able to handle 10x the number of requests and traffic (because the server isn't doing anything other than getting the request, and reading the page from memory into the TCP/IP buffer) but it will also give you access to metrics that might help you slow down bots. For instance, by correlating IP addresses you can simply block more than one connection per second per IP. Humans can't go faster than that, and even people using the same NATed IP address will only infrequently be blocked. You'd want to do a slow block - leave the connection alone for a full second before officially terminating the session. This can feed into a firewall to give longer term blocks to especially egregious offenders.

But the reality is that no matter what you do, there's no way to tell a human apart from a bot when the bot is custom built by a human for a single purpose. The bot is merely a proxy for the human.

Conclusion

At the end of the day, you can't tell a human and a computer apart for watching the front page. You can stop bots at the ordering step, but the bot users still have a first mover advantage, and you still have a huge load to manage.

You can add blocks for the simple bots, which will raise the bar and fewer people with bother with it. That may be enough.

But without changing your basic model, you're out of luck. The best you can do is take care of the simple cases, make the server so fast regular users don't notice, and sell so many items that even if you have a few million bots, as many regular users as want them will get them.

You might consider setting up a honeypot and marking user accounts as bot users, but that will have a huge negative community backlash.

Every time I think of a "well, what about doing this..." I can always counter it with a suitable bot strategy.

Even if you make the front page a captcha to get to the ordering page ("This item's ordering button is blue with pink sparkles, somewhere on this page") the bots will simply open all the links on the page, and use whichever one comes back with an ordering page. That's just no way to win this.

Make the servers fast, put in a reCaptcha (the only one I've found that can't be easily fooled, but it's probably way too slow for your application) on the ordering page, and think about ways to change the model slightly so regular users have as good a chance as the bot users.

-Adam

edited Jun 20, 2020 at 9:12

community wiki
2 revs
Adam Davis

"Every time I think of a "well, what about doing this..." I can always counter it with a suitable bot strategy" I came to the same conclusion when designing my authentication system, BUT -- there is one difference here that makes me doubt that logic: False positives aren't a big problem – Jens Roland Feb 8, 2009 at 18:11

(continued) E.g. if a few real users here and there are unable to get the special offers, that's actually not a big dealbreaker (as long as they don't know what they're missing). In an auth system, it *is* a dealbreaker - you don't want users being

prevented from logging in – Jens Roland Feb 8, 2009 at 18:13

(continued) What this means is, you can design the Woot system to be more restrictive than 'traditional' spambot countermeasures, and because of this, you may actually be able to thwart the bots effectively. – Jens Roland Feb 8, 2009 at 18:15

(however, now that I've given it some more thought, I can't think of a way that works, that will also thwart distributd / botnet 'attacks') – Jens Roland Feb 8, 2009 at 20:26

12

Disclaimer: This answer is completely non-programming-related. It does, however, try to attack the reason for scripts in the first place.

Another idea is if you truly have a limited quantity to sell, why don't you change it from a first-come-first-served methodology? Unless, of course, the hype is part of your marketing scheme.

There are many other options, and I'm sure others can think of some different ones:

- an ordering queue (pre-order system) - Some scripts might still end up at the front of the queue, but it's probably faster to just manually enter the info.
- a raffle system (everyone who tries to order one is entered into the system) - This way the people with the scripts have just the same chances as those without.

- a rush priority queue - If there is truly a high perceived value, people may be willing to pay more. Implement an ordering queue, but allow people to pay more to be placed higher in the queue.
- auction (credit goes to David Schmitt for this one, comments are my own) - People can still use scripts to snipe in at the last minute, but not only does it change the pricing structure, people are expecting to be fighting it out with others. You can also do things to restrict the number of bids in a given time period, make people phone in ahead of time for an authorization code, etc.

Share Improve this answer Follow

edited Jan 16, 2009 at 16:38

community wiki
3 revs
lc.

any raffle system will just be overloaded to increase the chances in the bot's favour – Andy Dent Feb 9, 2009 at 1:10

Not if you limit it to one per person/household/(physical) address it won't – lc. Feb 9, 2009 at 1:51

12

No matter how secure the Nazi's thought their communications were, the allies would often break their messages. No matter how you try to stop bots from using

your site the bot owners will work out a way around it. I'm sorry if that makes you the Nazi :-)

I think a different mindset is required

- Do not try to stop bots from using your site
- Do not go for a fix that works immediately, play the long game

Get into the mindset that it doesn't matter whether the client of your site is a human or a bot, both are just paying customers; but one has an unfair advantage over the other. Some users without much of a social life (hermits) can be just as annoying for your site's other users as bots.

Record the time you publish an offer and the time an account opts to buy it.

> This gives you a record of how quickly the client is buying stuff.

Vary the time of day you publish offers.

> For example, have a 3 hour window starting at some obscure time of the day (midnight?) Only bots and hermits will constantly refresh a page for 3 hours just to get an order in within seconds. Never vary the base time, only the size of the window.

Over time a picture will emerge.

01: You can see which accounts are regularly buying products within seconds of them going live. Suggesting they might be bots.

02: You can also look at the window of time used for the offers, if the window is 1 hour then some early buyers will be humans. A human will rarely refresh for 4 hours though. If the elapsed time is quite consistent between publish/purchase regardless of the window duration then that's a bot. If the publish/purchase time is short for small windows and gets longer for large windows, that's a hermit!

Now instead of stopping bots from using your site you have enough information to tell you which accounts are certainly used by bots, and which accounts are likely to be used by hermits. What you do with that information is up to you, but you can certainly use it to make your site fairer to people who have a life.

I think banning the bot accounts would be pointless, it would be akin to phoning Hitler and saying "Thanks for the positions of your U-boats!" Somehow you need to use the information in a way that the account owners wont realise. Let's see if I can dream anything up.....

Process orders in a queue:

When the customer places an order they immediately get a confirmation email telling them their order is placed in a

queue and will be notified when it has been processed. I experience this kind of thing with order/dispatch on Amazon and it doesn't bother me at all, I don't mind getting an email days later telling me my order has been dispatched as long as I immediately get an email telling me that Amazon knows I want the book. In your case it would be an email for

1. Your order has been placed and is in a queue.
2. Your order has been processed.
3. Your order has been dispatched.

Users think they are in a fair queue. Process your queue every 1 hour so that normal users also experience a queue, so as not to arouse suspicion. Only process orders from bot and hermit accounts once they have been in the queue for the "average human ordering time + x hours". Effectively reducing bots to humans.

edited Feb 7, 2009 at 11:01

community wiki
2 revs
Peter Morris

Ah thanks :-) I mention Nazi's because I am very interested in WWII stories about Bletchley park :-) Some of the stories on how messages were broken used a different mental approach to the problem, such as assuming operators were too lazy to change the codes from the night before :-)
– Peter Morris Feb 7, 2009 at 19:00

12

I say expose the price information using an API. This is the unintuitive solution but it does work to give you control over the situation. Add some limitations to the API to make it slightly less functional than the website.

You could do the same for ordering. You could experiment with small changes to the API functionality/performance until you get the desired effect.

There are proxies and botnets to defeat IP checks. There are captcha reading scripts that are extremely good. There are even teams of workers in India who defeat captchas for a small price. Any solution you can come up with can be reasonably defeated. Even Ned Batchelder's solutions can be stepped past by using a WebBrowser control or other simulated browser combined with a botnet or proxy list.

Share Improve this answer Follow

edited Jun 3, 2015 at 12:04

community wiki

5 revs, 2 users 92%
Robert Venables

9

We are currently using the latest generation of BigIP load balancers from F5 to do this. The BigIP has advanced traffic management features that can identify scrapersand bots based on frequency and patterns of use even from amongst a set of sources behind a single IP. It can then throttle these, serve them alternative content or simply tag them with headers or cookies so you can identify them in your application code.

Share Improve this answer Follow

answered Feb 7, 2009 at 9:42

community wiki
ozy

This is the exact solution I was going to suggest, especially the automatic throttling. You could roll your own, just relies on some regular to advanced signal analysis. – wds Feb 7, 2009 at 18:58

7

How about introducing a delay which requires human interaction, like a sort of "CAPTCHA game". For example, it could be a little Flash game where during 30 seconds they have to burst checkered balls and avoid bursting solid balls (avoiding colour blindness issues!). The game would be given a random number seed and what the

game transmits back to the server would be the coordinates and timestamps of the clicked points, along with the seed used.

On the server you simulate the game mechanics using that seed to see if the clicks would indeed have burst the balls. If they did, not only were they human, but they took 30 seconds to validate themselves. Give them a session id.

You let that session id do what it likes, but if makes too many requests, they can't continue without playing again.

Share Improve this answer Follow

answered Jan 16, 2009 at 16:17

community wiki
Paul Dixon

Fun idea, but totally and completely ruining the user experience. Normal people visiting the site will think of it as 30 seconds of useless waiting. 30 seconds of useless waiting when browsing the internet or using web-apps is not in any way acceptable. – Arve Systad Feb 8, 2009 at 2:31

normal people visiting wouldn't trigger the delay, only someone making an unreasonable number of requests. The idea *is* a little tongue in cheek, but I can see it working if the target audience are used to little flash games :) – Paul Dixon Feb 8, 2009 at 12:45

Entertaining (and nigh-foolproof) idea, but I'd be irritated (especially during a Bag Of Canaries frenzy), and that would require massively more processing on their servers to

perform checking (which is a big part of the problem). Also, bots can burst bubbles. You'd have to frequently change rules. – Groxx Feb 9, 2009 at 23:53

Assuming each game is issued a token, and you know the time you issued the tokens, you need only attempt to process a token once, and only between 30 and say 300 seconds after it was issued. The beauty of it is that even if a bot does burst the bubble, they've still waited 30 seconds to do so. – Paul Dixon Feb 10, 2009 at 9:54

Plus, let's not forget the idea is to limit traffic. The page could say "we're very busy, if you're in a hurry, play this game for 30 seconds, or try again in a few minutes... – Paul Dixon Feb 10, 2009 at 9:56

7

First, let me recap what we need to do here. I realize I'm just paraphrasing the original question, but it's important that we get this 100% straight, because there are a lot of great suggestions that get 2 or 3 out of 4 right, but as I will demonstrate, you will need a multifaceted approach to cover all of the requirements.

Requirement 1: Getting rid of the 'bot slamming':

The rapid-fire 'slamming' of your front page is hurting your site's performance and is at the core of the problem. The 'slamming' comes from both single-IP bots and - supposedly - from botnets as well. We want to get rid of both.

Requirement 2: Don't mess with the user experience:

We could fix the bot situation pretty effectively by implementing a nasty verification procedure like phoning a human operator, solving a bunch of CAPTCHAs, or similar, but that would be like forcing every innocent airplane passenger to jump through crazy security hoops just for the slim chance of catching the very stupidest of terrorists. Oh wait - we actually do that. But let's see if we can *not* do that on woot.com.

Requirement 3: Avoiding the 'arms race':

As you mention, you don't want to get caught up in the spambot arms race. So you can't use simple tweaks like hidden or jumbled form fields, math questions, etc., since they are essentially obscurity measures that can be trivially autodetected and circumvented.

Requirement 4: Thwarting 'alarm' bots:

This may be the most difficult of your requirements. Even if we can make an effective human-verification challenge, bots could still poll your front page and alert the scripter when there is a new offer. We want to make those bots infeasible as well. This is a stronger version of the first requirement, since not only can't the bots issue performance-damaging rapid-fire requests -- they can't even issue enough repeated requests to send an 'alarm' to the scripter in time to win the offer.

Okay, so let's se if we can meet all four requirements. First, as I mentioned, no one measure is going to do the

trick. You will have to combine a couple of tricks to achieve it, and you will have to swallow two annoyances:

1. A small number of users will be required to jump through hoops
2. A small number of users will be unable to get the special offers

I realize these are annoying, but if we can make the 'small' number *small enough*, I hope you will agree the positives outweigh the negatives.

First measure: User-based throttling:

> This one is a no-brainer, and I'm sure you do it already. If a user is logged in, and keeps refreshing 600 times a second (or something), you stop responding and tell him to cool it. In fact, you probably throttle his requests significantly sooner than that, but you get the idea. This way, a logged-in bot will get banned/throttled as soon as it starts polling your site. This is the easy part. The unauthenticated bots are our real problem, so on to them:

Second measure: Some form of IP throttling, as suggested by nearly everyone:

> No matter what, you will have to do *some* IP based throttling to thwart the 'bot slamming'.

> Since it seems important to you to allow unauthenticated (non-logged-in) visitors to get the special offers, you only have IPs to go by initially, and although they're not perfect, they *do* work against single-IP bots. Botnets are a different beast, but I'll come back to those. For now, we will do some simple throttling to beat rapid-fire single-IP bots.
>
> The performance hit is negligable if you run the IP check before all other processing, use a proxy server for the throttling logic, and store the IPs in a memcached lookup-optimized tree structure.

Third measure: Cloaking the throttle with cached responses:

> With rapid-fire single-IP bots throttled, we still have to address slow single-IP bots, ie. bots that are specifically tweaked to 'fly under the radar' by spacing requests slightly further apart than the throttling prevents.
>
> To instantly render slow single-IP bots useless, simply use the strategy suggested by abelenky: serve 10-minute-old cached pages to all IPs that have been spotted in the last 24 hours (or so). That way, every IP gets one 'chance' per day/hour/week (depending on the period you choose), and there will be no visible annoyance

> to real users who are just hitting 'reload', except that they don't win the offer.
>
> The beauty of this measure is that is **also** thwarts 'alarm bots', as long as they don't originate from a botnet.
>
> (I know you would probably prefer it if real users were allowed to refresh over and over, but there is no way to tell a refresh-spamming human from a request-spamming bot apart without a CAPTCHA or similar)

Fourth measure: reCAPTCHA:

> You are right that CAPTCHAs hurt the user experience and should be avoided. However, in _*one*_ situation they can be your best friend: If you've designed a very restrictive system to thwart bots, that - because of its restrictiveness - also catches a number of false positives; then a CAPTCHA served *as a last resort* will allow those real users who get caught to slip by your throttling (thus avoiding annoying DoS situations).
>
> The sweet spot, of course, is when ALL the bots get caught in your net, while extremely few real users get bothered by the CAPTCHA.

> If you, when serving up the 10-minute-old cached pages, also offer an alternative, *optional*, CAPTCHA-verified 'front page refresher', then humans who **really** want to keep refreshing, can still do so without getting the old cached page, but at the cost of having to solve a CAPTCHA for each refresh. That *is* an annoyance, **but an optional one** just for the die-hard users, who tend to be more forgiving because they *know* they're gaming the system to improve their chances, and that improved chances don't come free.

Fifth measure: Decoy crap:

> Christopher Mahan had an idea that I rather liked, but I would put a different spin on it. Every time you are preparing a new offer, prepare two other 'offers' as well, that no human would pick, like a 12mm wingnut for $20. When the offer appears on the front page, put all three 'offers' in the same picture, with numbers corresponding to each offer. When the user/bot actually goes on to order the item, they will have to pick (a radio button) which offer they want, and since most bots would merely be guessing, in two out of three cases, the bots would be buying worthless junk.

> Naturally, this doesn't address 'alarm bots', and there is a (slim) chance that someone could build a bot that was able to pick the correct item. However, the risk of accidentally buying junk should make scripters turn entirely from the fully automated bots.

Sixth measure: Botnet Throttling:

> [deleted]

Okay..... I've now spent most of my evening thinking about this, trying different approaches.... global delays.... cookie-based tokens.. queued serving... 'stranger throttling'.... And it just doesn't work. It doesn't. I realized the main reason why you hadn't accepted any answer yet was that noone had proposed a way to thwart a distributed/zombie net/botnet attack.... so I really wanted to crack it. I believe I cracked the botnet problem for authentication in a different thread, so I had high hopes for your problem as well. But my approach doesn't translate to this. You only have IPs to go by, and a large enough botnet doesn't reveal itself in any analysis based on IP addresses.

So there you have it: My sixth measure is naught. Nothing. Zip. Unless the botnet is small and/or fast enough to get caught in the usual IP throttle, I don't see ***any*** effective measure against botnets that doesn't involve explicit human-verification such as CAPTHAs. I'm

sorry, but I think combining the above five measures is your best bet. And you could probably do just fine with just abelenky's 10-minute-caching trick alone.

Share Improve this answer

Follow

edited May 23, 2017 at 12:18

Community Bot

1 ● 1

answered Feb 8, 2009 at 20:45

Jens Roland

27.8k ● 15 ● 84 ● 105

doesn't 3. mean you're serving up old pages to all of AOL, assuming a few bots come from AOL's IP pool? – Andy Dent Feb 9, 2009 at 1:18

@Andy: Only if *all* AOL users share the same IP addresses that the bots used while spamming. – Jens Roland Feb 9, 2009 at 10:19

6

There are a few other / better solutions already posted, but for completeness, I figured I'd mention this:

If your main concern is performance degradation, and you're looking at true **hammering**, then you're actually dealing with a DoS attack, and you should probably try to handle it accordingly. One common approach is to simply drop packets from an IP in the firewall after a number of connections per second/minute/etc. For example, the standard Linux firewall, iptables, has a standard operation matching function 'hashlimit', which could be used to

correlate connection requests per time unit to an IP-address.

Although, this question would probably be more apt for the next SO-derivate mentioned on the last SO-podcast, it hasn't launched yet, so I guess it's ok to answer :)

EDIT:
As pointed out by novatrust, there are still ISPs actually NOT assigning IPs to their customers, so effectively, a script-customer of such an ISP would disable all-customers from that ISP.

Share Improve this answer Follow

edited Jan 16, 2009 at 16:28

community wiki
2 revs
roe

Unfortunately some ISPs have shared exit IP addresses. For example, AOL has a limited collection of IPs that the members appear under: webmaster.info.aol.com/proxyinfo.html Your solution would impose a hard limit on the number of users for many ISPs. – Robert Venables Jan 16, 2009 at 16:21

Wow, I am awestruck. Stuff like this is still going on? – falstro Jan 16, 2009 at 16:26

Holy cow. I guess AOL won't be accessing my site then. – Karl Jan 16, 2009 at 16:30

1. Provide an RSS feed so they don't eat up your bandwidth.
2. When buying, make everyone wait a **random** amount of time of up to 45 seconds or something, depending on what you're looking for exactly. Exactly what are your timing constraints?
3. Give everyone 1 minute to put their name in for the drawing and then randomly select people. I think this is the fairest way.
4. Monitor the accounts (include some times in the session and store it?) and add delays to accounts that seem like they're below the human speed threshold. That will at least make the bots be programmed to slow down and compete with humans.

edited Jan 16, 2009 at 16:48

community wiki
3 revs
Joe Phillips

These are interesting concepts but the "random selection" and the waiting period removes much of the "frenzy" that I am guessing woot is depending on. Taking away the timing urgency kind've ruins the site. – TM. Feb 7, 2009 at 21:00

If it looks like a drawing, then he has to deal with gambling laws. Not worth it. – jmucchiello Feb 11, 2009 at 23:33

5

First of all, by definition, it is impossible to support stateless, i.e. truly anonymous, transactions while also being able to separate the bots from legitimate users.

If we can accept a premise that we can impose some cost on a brand-spanking-new woot visitor on his first page hit(s), I think I have a possible solution. For lack of a better name, I'm going to loosely call this solution "A visit to the DMV."

Let's say that there's a car dealership that offers a different new car each day, and that on some days, you can buy an exotic sports car for $5 each (limit 3), plus a $5 destination charge.

The catch is, the dealership requires you to visit the dealership and show a valid driver's license before you're allowed in through the door to see what car is on sale. Moreover, you must have said valid driver's license in order to make the purchase.

So, the first-time visitor (let's call him Bob) to this car dealer is refused entry, and is referred to the DMV office (which is conveniently located right next door) to obtain a driver's license.

Other visitors with a valid driver's license is allowed in, after showing his driver's license. A person who makes a nuisance of himself by loitering around all day, pestering the salesmen, grabbing brochures, and emptying the

complimentary coffee and cookies will eventually be turned away.

Now, back to Bob without the license -- all he has to do is endure the visit to the DMV once. After that, he can visit the dealership and buy cars anytime he likes, unless he accidentally left his wallet at home, or his license is otherwised destroyed or revoked.

The driver's license in this world is nearly impossible to forge.

The visit to the DMV involves first getting the application form at the "Start Here" queue. Bob has to take the completed application to window #1, where the first of many surly civil servants will take his application, process it, and if everything is in order, stamp the application for the window and send him to the next window. And so, Bob goes from windows to window, waiting for each step of his application to go through, until he finally gets to the end and receives his drivere's license.

There's no point in trying to "short circuit" the DMV. If the forms are not filled out correctly in triplicate, or any wrong answers given at any window, the application is torn up, and the hapless customer is sent back to the start.

Interestingly, no matter how full or empty the office is, it takes about the same amount of time to get serviced at each successive window. Even when you're the only person in line, it seems that the personnel likes to make

you wait a minute behind the yellow line before uttering, "Next!"

Things aren't quite so terrible at the DMV, however. While all the waiting and processing to get the license is going on, you can watch a very entertaining and informative infomercial for the car dealership while you're in the DMV lobby. In fact, the infomerical runs just long enough to cover the amount of time you spend getting your license.

The slightly more technical explanation:

As I said at the very top, it becomes necessary to have some statefulness on the client-server relationship which allows you to separate humans from bots. You want to do it in a way that doesn't overly penalize the anonymous (non-authenticated) human visitor.

This approach probably requires an AJAX-y client-side processing. A brand-spanking-new visitor to woot is given the "Welcome New User!" page full of text and graphics which (by appropriate server-side throttling) takes a few seconds to load completely. While this is happening (and the visitor is presumably busy reading the welcome page(s)), his identifying token is slowly being assembled.

Let's say, for discussion, the token (aka "driver's license) consists of 20 chunks. In order to get each successive chunk, the client-side code must submit a valid request to the server. The server incorporates a deliberate delay (let's say 200 millisecond), before sending the next chunk along with the 'stamp' needed to make the next chunk

request (i.e., the stamps needed to go from one DMV window to the next). All told, about 4 seconds must elapse to finish the chunk-challenge-response-chunk-challenge-response-...-chunk-challenge-response-completion process.

At the end of this process, the visitor has a token which allows him to go to the product description page and, in turn, go to the purchasing page. The token is a unique ID to each visitor, and can be used to throttle his activities.

On the server side, you only accept page views from clients that have a valid token. Or, if it's important that everyone can ultimately see the page, put a time penalty on requests that is missing a valid token.

Now, for this to be relatiely benign to the legitimate human visitor,t make the token issuing process happen relatively non-intrusively in the background. Hence the need for the welcome page with entertaining copy and graphics that is deliberately slowed down slightly.

This approach forces a throttle-down of bots to either use an existing token, or take the minimum setup time to get a new token. Of course, this doesn't help as much against sophisticated attacks using a distributed network of faux visitors.

answered Feb 7, 2009 at 10:15

community wiki
Toybuilder

5

Write a reverse-proxy on an apache server in front of your application which implements a Tarpit (Wikipedia Article) to punish bots. It would simply manage a list of IP addresses that connected in the last few seconds. You detect a burst of requests from a single IP address and then exponentially delay those requests before responding.

Of course, multiple humans can come from the same IP address if they're on a NAT'd network connection but it's unlikely that a human would mind your response time going for 2mS to 4mS (or even 400mS) whereas a bot will be hampered by the increasing delay pretty quickly.

Share Improve this answer Follow

answered Feb 8, 2009 at 1:45

community wiki
Denis Hennessy

4

I'm not seeing the great burden that you claim from checking incoming IPs. On the contrary, I've done a project for one of my clients which analyzes the HTTP access logs every five minutes (it could have been real-time, but he didn't want that for some reason that I never fully understood) and creates firewall rules to block connections from any IP addresses that generate an

excessive number of requests unless the address can be confirmed as belonging to a legitimate search engine (google, yahoo, etc.).

This client runs a web hosting service and is running this application on three servers which handle a total of 800-900 domains. Peak activity is in the thousand-hits-per-second range and there has never been a performance issue - firewalls are very efficient at dropping packets from blacklisted addresses.

And, yes, DDOS technology definitely does exist which would defeat this scheme, but he's not seeing that happen in the real world. On the contrary, he says it's vastly reduced the load on his servers.

Share Improve this answer Follow

answered Jan 16, 2009 at 17:13

community wiki
Dave Sherohman

4

My approach would be to focus on non-technological solutions (otherwise you're entering an arms race you'll lose, or at least spend a great deal of time and money on). I'd focus on the billing/shipment parts - you can find bots by either finding multiple deliveries to same address or by multiple charges to a single payment method. You can even do this across items over several weeks, so if a user got a previous item (by responding really really fast)

he may be assigned some sort of "handicap" this time around.

This would also have a side effect (beneficial, I would think, but I could be wrong marketing-wise for your case) of perhaps widening the circle of people who get lucky and get to purchase woot.

Share Improve this answer Follow

answered Feb 7, 2009 at 9:49

community wiki
Shachar

4

You can't totally prevent bots, even with a captcha. However you can make it a pain to write and maintain a bot and therefore reduce the number. Particularly by forcing them to update their bots daily you'll be causing most to lose interest.

Here are a some ideas to make it harder to write bots:

- Require running a javascript function. Javascript makes it much more of a pain to write a bot. Maybe require a captcha if they aren't running javascript to still allow actual non-javascript users (minimal).
- Time the keystrokes when typing into the form (again via javascript). If it's not human-like then reject it. It's a pain to mimic human typing in a bot.

- Write your code to update your field ID's daily with a new random value. This will force them to update their bot daily which is a pain.
- Write your code to re-order your fields on a daily basis (obviously in some way that's not random to your users). If they're relying on the field order, this will trip them up and again force daily maintenance to their bot code.
- You could go even further and use Flash content. Flash is totally a pain to write a bot against.

Generally if you start taking a mindset of not preventing them, but making it more work for them, you can probably achieve the goal you're looking for.

answered Feb 7, 2009 at 18:15

community wiki
jwanagel

Humans sometimes engage in non-human typing, though--form fillers. – Loren Pechtel Feb 7, 2009 at 19:57

You need to allow for very different typing styles/speeds - everything from hunt'n'peck to touchtyping. It's not hard to write bot that falls somewhere between. Things like variable field IDs and order can be circumvented by reading and parsing of form, which is not very hard. – Kornel Feb 16, 2009 at 10:03

4

Stick a 5 minute delay on all product announcements for unregistered users. Casual users won't really notice this and noncasual users will be registered anyhow.

Share Improve this answer Follow

answered Feb 9, 2009 at 14:43

community wiki
Brian

3

Time-block user agents that make so-many requests per minute. Eg if you've got somebody requesting a page exactly every 5 seconds for 10 minutes, they're probably not a user... But it could be tricky to get this right.

If they trigger an alert, redirect every request to a static page with as little DB-IO as possible with a message letting them know they'll be allowed back on in X minutes.

It's important to add that you should probably only apply this on requests for pages and ignore all the requests for media (js, images, etc).

Share Improve this answer Follow

answered Jan 16, 2009 at 16:12

community wiki
Oli

I've done this on a personal project, it seems like a good method. You just need to remember all of the ip's as they hit your page, and have rules set up for what it means to be hitting your page too often. The problem is that the OP said checking IPs is way too expensive, which I dont understand. – Karl Jan 16, 2009 at 16:27

If you implement the IP checking yourself (i.e. in your database, from your PHP script or whatever), then it will be quite expensive. Get the firewall to do it for you and it becomes much more feasible. – rmeador Jan 16, 2009 at 18:26

rmeador: It also seems like it would be a lot harder to determine if the request was for HTML or other media. If you've got 20 external things on your page, you're looking at a minimum of 21 requests for a new user in 1-2 seconds. – Oli Jan 16, 2009 at 20:00

3

Preventing DoS would defeat #2 of @davebug's goals he outlined above, "Keep the site at a speed not slowed by bots" but wouldn't necessary solve #1, "Sell the item to non-scripting humans"

I'm sure a scripter could write something to skate just under the excessive limit that would still be faster than a human could go through the ordering forms.

Share Improve this answer Follow

answered Jan 16, 2009 at 17:35

community wiki
Shawn Miller

3

All right so the spammers are out competing regular people to win the "bog of crap" auction? Why not make the next auction be a literal "bag of crap"? The spammers get to pay good money for a bag full of doggy do, and we all laugh at them.

Share Improve this answer Follow

answered Feb 7, 2009 at 7:52

community wiki
1800 INFORMATION

3

The important thing here is to change the system to remove load from your server, prevent bots from winning the bag of crap WITHOUT letting the botlords know you are gaming them or they will revise their strategy. I don't think there is any way to do this without some processing at your end.

So you record hits on your home page. Whenever someone hits the page that connection is compared to its last hit, and if it was too quick then it is sent a version of the page without the offer. This can be done by some sort of load balancing mechanism that sends bots (the hits that are too fast) to a server that simply serves cached versions of your home page; real people get sent to the good server. This takes the load off the main server and

makes the bots think that they are still being served the pages correctly.

Even better if the offer can be declined in some way. Then you can still make the offers on the faux server but when the bot fills out the form say "Sorry, you weren't quick enough" :) Then they will definitely think they are still in the game.

Share Improve this answer Follow

answered Feb 7, 2009 at 11:19

community wiki
Chris Latta

3

Most purely technical solutions have already been offered. I'll therefore suggest another view of the problem.

As I understand it, the bots are set up by people **genuinely** trying to buy the bags you're selling. The problem is -

1. Other people, who don't operate bots, deserve a chance to buy, and you're offering a limited amount of bags.
2. You want to attract humans to your site and just sell the bags.

Instead of trying to avoid the bots, you can enable potential bag-buyers to subscribe to an email, or even

SMS update, to get notified when a sell will take place. You can even give them a minute or two head start (a special URL where the sell starts, randomly generated, and sent with the mail/SMS).

When these buyers go to buy they're in you're site, you can show them whatever you want in side banners or whatever. Those running the bots will prefer to simply register to your notification service.

The bots runners might still run bots on your notification to finish the buy faster. Some solutions to that can be offering a one-click buy.

By the way, you mentioned your users are not registered, but it sounds like those buying these bags are not random buyers, but people who look forward to these sales. As such, they might be willing to register to get an advantage in trying to "win" a bag.

In essence what I'm suggesting is try and look at the problem as a social one, rather than a technical one.

Asaf

edited Feb 7, 2009 at 22:18

answered Feb 7, 2009 at 18:47

Asaf R
6,978 ● 9 ● 49 ● 70

2

You could try to make the price harder for scripts to read. This is achieved most simply by converting it to an image, but a text recognition algorithm could still get around this. If enough scripters get around it, you could try applying captcha-like things to this image, but obviously at the cost of user experience. Instead of an image, the price could go in a flash app.

Alternately, you could try to devise a way to "shuffle" the HTML pf a page in some way that doesn't affect the rendering. I can't think of a good example off the top of my head, but I'm sure it's somehow doable.

Share Improve this answer Follow

answered Jan 16, 2009 at 16:00

community wiki
Matt Boehm

2

How about this: Create a form to receive an email if a new item is on sale and add a catching system that will serve the same content to anyone refreshing in less than X seconds.

This way you win all the escenarios: you get rid of the scrapers(they can scrape their email account) and you give chance to the people who wont code something just to buy in your site! Im sure i would get the email in my mobile and log in to buy something if i really wanted to.

Share Improve this answer Follow

answered Feb 9, 2009 at 18:49

community wiki
DFectuoso