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)

0

I am not 100% sure this would work, at least not without trying.

But it seems as if it should be possible, although technically challenging, to write a server-side HTML/CSS scrambler that takes as its input a normal html page + associated files, and outputs a more or less blank html page, along with an obfuscated javascript file that is capable of reconstructing the page. The javascript couldn't just print out straightforward DOM nodes, of course... but it could spit out a complex set of overlapping, absolute-positioned divs and paragraphs, each containing one letter, so it comes out perfectly readable.

Bots won't be able to read it unless they have employ a complete rendering engine and enough AI to reconstruct what a human would be seeing.

Then, because it's an automated process, you can re-scramble the site as often as you have the computational power for - every minute, or every ten minutes, or every hour, or even every page load.

Granted, writing such an obfuscater would be difficult, and probably not worth it. But it's a thought.

Share Improve this answer Follow

answered Feb 9, 2009 at 18:13

community wiki
levand

0

There's a lot of suggestions here so pardon me if this has already been posted.

The first thing I would do is make the ordering a two step process. The first step would pass back a GUID while logging the IP Address. The second step would receive the GUID and compare it against IP Addresses that have been logged. In conjunction with blocking IP Addresses which are spamming the site (IE: faster than a human can click refresh) this technique could stop spammers from successfully making purchases thereby solving 1 & 3.

The second item is problematic but I would keep a running list of your regular user's IP addresses and throttle traffic for any newcomers. This could leave first time visitors and dial up users (due to changing IP addresses) out in the cold, but I think it's just making the best out of a bad situation by giving preference to repeat business... and dialup users, well it's questionable whether they'd "win" even if there weren't any spammers anyway.

Share Improve this answer Follow

answered Feb 10, 2009 at 8:47

community wiki
Spencer Ruport

0

- Go after the money stream. It is much easier than tracking the IP side. Make bots pay too much a few times (announcement with white text on white background and all variants of it) kills their business case quickly. You should prepare this carefully, and make good use of the strong points of bots: their speed. Did you try a few thousand fake announcements a few seconds apart? If they are hitting ten times/second you can go even faster. You want to keep this up as long as they keep buying, so think carefully about the moment of the day/week you want to start this. Ideally, they will stop paying, so you can hand over your case to a bank.
- Make sure your site is fully generated, and each page access returns different page content (html, javascript and css). Parsing is more difficult than generating, and it is easy to build-in more variation than bot developers can handle. Keep on changing the content and how you generate it.
- You need to know how fast bots can adapt to changes you make, and preferably the timezone they are in. Is it one botnet or more, are they in the same timezone, a different one, or is it a worldwide

developer network? You want your counterattack to be timed right.

- Current state of the art bots have humans enter captcha's (offered against porn/games).
- Make it unattractive to react very fast.
- Use hashes and honeypots, as Ned Batchelder explains.

[edit] It is simply not true that you cannot defend against botnets. Especially my second suggestion provides for adequate defense against automated buyers. it requires a complete rethinking about the technology you're using, though. You might want to do some experiments with Seaside, or alternatively directly in c.

Share Improve this answer Follow

edited Feb 10, 2009 at 18:41

community wiki
2 revs
Stephan Eggermont

0

Why don't you block the credit cards of users you identify as bots?

1. Publish that using bots is illegal on your website
2. Find certain heuristics that identify bots (this can be done for example by short-term IP tracking or by the time it takes them to feel up the form)

3. If someone you tagged as a bot purchased the item, block his credit card for future use
4. Next time he tries to make a purchase, disallow it and return the item to stock

I guess even the professionals will run out of credit cards eventually.

Your server load should decrease with time once the botters give up on you. Another idea is to separate your pages between servers - e.g., RSS feed on one server, homepage on another, checkout on another one.

Good luck.

Share Improve this answer Follow

answered Feb 10, 2009 at 22:10

community wiki
idophir

> Well... good point... I guess they will get tired of having to open a new PP account every day at some point. As opposed to all other suggestions, my approach is trying to hit the scripters in a place they cannot overcome by some code improvement. It's not perfect but I think it's cost/effective.
> – idophir Feb 11, 2009 at 20:06

Trying to target the BOTs themselves will never solve the problem - whoever is writing them will figure out a new way around whatever you've put in place. However

0

forcing the user to think before buying would be a much more effective solution. The best way of doing this that I can think of is run a Dutch auction. Start the price high (2x what you buy it for in the shop) and decrease it over time. The first person to hit buy gets it. I don't think any bot is intelligent enough to workout what the best price is for the item.

Share Improve this answer Follow

answered Feb 11, 2009 at 21:26

community wiki
Wairapeti

0

Restrict the times at which you release offers: For example: only from 7 minutes to 8 minutes past the start of an hour. Do not deviate from this, and give penalties on the order of a couple seconds to IPs which check a lot in the half hour before the release time. It then becomes advantageous for bot owners to only screen scrape for a couple minutes every hour instead of all. the. time. Also, because a normal person can check a site once every hour but not every second, you put normal people on a much more even footing with the bots.

Cookies: Use a tracking cookie composed of only a unique ID (a key for a database table). Give "release delays" to clients with no cookie, invalid cookies, clients which use the same cookie from a new IP, or cookies used with high frequency.

Identify likely bots: Cookies will cause the bots to request multiple cookies for each IP they control, which is behavior which can be tracked. IPs with only a single issued cookie are most likely normal clients. IPs with many issued cookies are either large NAT-ed networks, or a bot. I'm not sure how you would distinguish those, but companies are probably more likely to have things like DNS servers, a web page, and things of that nature.

Share Improve this answer Follow

answered Feb 12, 2009 at 18:45

community wiki
Craig Gidney

0

Perhaps you need a solution that makes it totally impossible for a bot to distinguish between the bag-o-crap sales and all other content.

This is sort of a variation on the captcha theme, but instead of the user authenticating themselves by solving the captcha, the captcha is instead the description of the sale, rendered in a visually pleasing (but perhaps somewhat obscured by the background) manner.

Share Improve this answer Follow

answered Feb 12, 2009 at 18:55

community wiki
SingleNegationElimination

0

I think your best bet is to watch IP's coming in, but to mitigate the issues you mention in a couple of ways. First, use a probabilistic hash (eg, a Bloom Filter) to mark IP's which have been seen before. This class of algorithm is very fast, and scales well to absolutely massive set sizes. Second, use a graduated response, whereby a server delay is added to each request, predicated by how much you've seen the IP 'recently'.

Share Improve this answer Follow

answered Feb 12, 2009 at 21:24

community wiki
Johnny Graettinger

0

At the expense of Usability by those with screen readers you could just, on 90% of the pages use unlabelled, undenotable picture buttons. Rotate the pictures regularly and use a random generator and random sorting to lay out two buttons that say "I want this" and "I am a bot". Place them side by sort in a different order. At each stage a user can make progress torwards their target but a bot is more likely to make a mistake (50% * number of steps).

It's like a capture at every stage on easier for the user and slower for bots who need to prompt their master at EVERY single step. Put the price, the confirm button, the item description in pictures. It sucks but likely more successful.

Share Improve this answer Follow

answered Feb 12, 2009 at 21:47

community wiki
Philluminati

0

Just make the bots compete on even ground. Encrypt a timestamp and stick it in a hidden form field. When you get a submission decrypt it and see how much time has passed. If it surpasses the threshold of human typing ability reject it. Now bots and humans can only try to buy the bag of crap at the same speed.

Share Improve this answer Follow

answered Feb 12, 2009 at 22:05

community wiki
Jason Christa

0

If you can't beat them... Change the rules!

Why not provide a better system than the scripters have made for themselves?

Modify your site to be fairer for people not using bot scripts. People register (CAPTCHA or email verification) and effectively enter a lottery competition to win!

'Winning' makes it more fun. and each person pays a small entry fee so the Winner gets the product for EVEN less

Share Improve this answer Follow

answered Feb 12, 2009 at 22:22

community wiki
Andrew Harry

0

I'm not a web developer, so take this with a pinch of salt, but here's my suggestion -

Each user has a cookie (containing a random string of data) that determines whether they see the current crap sale.

(If you don't have a cookie, you don't see them. So users who don't enable cookies never see crap sales; and a new user will never see them the first time they view the page, but will thereafter).

Each time the user refreshes the website, he passes his current cookie to the server, and the server uses that to decide whether to give him a new cookie or leave the current one unchanged; and based on that, decides whether to show the page with or without the crap sale.

To keep things simple on the server side, you could say at any given time, there's only ever one cookie that will let you see crap sales; and there are a couple of other cookies that are labelled "generated in the last 2 seconds", which will always be kept unchanged. So if you refresh the page faster than that, you can't get a new one.

(...ah, well, I guess that doesn't stop a bot from restoring an older cookie and passing it back to you. Still, maybe there's a solution here somewhere.)

Share Improve this answer Follow

answered Feb 12, 2009 at 22:44

community wiki
Laurie Cheers

0

Stopping all bots would be quite difficult, especially without using a CAPTCHA. I think you should approach this from the standpoint of implementing a wide variety of measures to make life harder for the scripters.

I believe this is one measure that would weed out some of them:

You could try **randomizing the IDs and class names** of your tags with each response. This would force bots to rely on the position and context of important tags, which requires a more sophisticated bot. Furthermore, you

could randomize the position of the tags if you want to use relative or absolute positioning in your CSS.

The biggest drawback with this approach is that you would have to take steps to ensure your CSS file is not cached client-side, because it would of course need to contain the randomized IDs & class names. One way to overcome this is to not use external CSS files and instead put the CSS with the randomized selectors in the `<head>` `</head>` section of the page. This would allow the randomized CSS to be client-side cached along with the rest of the page.

answered Feb 12, 2009 at 22:45

community wiki
Seibar

0

How about coming up with a way to identify bots, probably IP based, but not block them from accessing the site, just don't allow them to actually buy anything. That is, if they buy, they don't actually get it, since bots are against the terms of use.

Share Improve this answer Follow

answered Feb 12, 2009 at 22:54

community wiki
David Durham

0

Steps:

(combining ideas from another poster and gif spammers)

- Display the **entire offer page as an image**, ad-copy and all.
- **Encrypt the price in the URL**.

Attacks:

1. **Bots going to the URL to view the price on the checkout page**
    - **turn the checkout price tag into an image**, or
    - **apply a captcha** before users can go to the order page.

2. **chewing up bandwidth**
    - Serve special offers using images, **normal offers using HTML**.
3. **reckless bot ordering**
    - some of the special "image" offers are actually at normal prices.
4. **RSS Scraping**
    - **RSS feeds must be paid for** by hashcash or captchas.
    - This has to be on a **per-request** basis.
    - It can be **pre-paid**, for instance user can enter 20 captchas for 200 RSS look ups
    - Once the threat of DDOS has been mitigated, you can **implement e-mail notification** of offers

edited Feb 12, 2009 at 23:08

community wiki
2 revs
Chui Tey

0

The problem with CAPTCHA is that when you see a crap sale on Woot, you have to act VERY fast as a consumer if you hope to receive your bag of crap. So, if you are going to use a form of CAPTCHA , it must be very quick for the customer.

What if you had a large image, say 600 x 600 that was just a white background and dots of different colors or patterns randomly placed on the image. The image would have an image map on it. This map would have a link mapped to small chunks of the image. Say, 10 x 10 blocks. The user would simply have to click on the specific type of dot. It would be quick for end the user and it would somewhat difficult for a bot developer to code. But this alone may not be that difficult for a good bot creator to get past. I would add ciphered URLs.

I was developing a system some time back that would cipher URLs. If every URL on these pages is ciphered with a random IV, Then they all appear to be unique to the bot. I was designing this to confuse probing bots. I have not completed the technique yet, but I did have a small site coded that functioned in this manor.

While these suggestions are not a full solution, they would make it way harder to build a working bot while still being easy for a human to use.

Share Improve this answer Follow

answered Feb 12, 2009 at 23:19

community wiki
Konrad

0

There's probably no good solution as long as the surprise distribution of the bag o' crap is tied only to a point in time - since bots have plenty of time, and the resources to keep slamming the site at short time intervals.

I think you'd have to add an extra criterion that bots can't screen-scrape or manipulate from their end. For instance, say at any time there's 5000 humans hitting the page a few times a minute looking for the bag of crap, and 50 bots slamming it every second. In the first few seconds after it appears, the 50 bots are going to snap it all up.

So, you could add a condition that the crap appears first to any users where the modulus 30 of their integer IP is a random number, say 17. Maybe another random number is added every second, so the crap is revealed incrementally to all clients over 30 seconds.

Now imagine what happens in the first several seconds: currently, all 50 bots are able to snap up all the crap immediately, and the humans get 0. Under this scheme, after 6 seconds only 10 bots have made it through, while 1000 humans have gotten through, and most of the crap goes to the humans. You could play with the timings and the random modulus to try and optimize that interval, depending on user counts and units available.

Not a perfect solution, but an improvement. The upside is many more humans than bots will benefit. There are several downsides, mainly that not every human gets an equal shot at the crap on any particular day - though they don't have much of a shot now, and I'd guess even without bots, most of them get shut out at random unless they happen to refresh at just the right second. And, it wouldn't work on a botnet with lots of distributed IPs. Dunno if anyone's really using a botnet just for woot crap though.

Share Improve this answer Follow

answered Feb 12, 2009 at 23:46

community wiki
Dogwelder

0

Your end goal is to spread out to a larger user base who gets to buy stuff.

What if you did something like releasing your bags of w00t over a period of an hour or two, and over a range of IP addresses, instead of releasing them all at the same time and to any IP address.

Let's say you have 255 bags of w00t. 1.0.0.0 can buy in the first minute, 2.0.0.0 can buy in the second minute (potentially 2 bags of w00t available), etc, etc.

Then, after 255 minutes, you have made bags of w00t available to everybody, although it is highly likely that not

all 255 bags of w00t are left.

This limits a true attack to users who have >255 computers, although a bot user might be able to "own" the bag of w00t assigned to their IP range.

There is no requirement that you match up bags to IP's fairly (and you definitely should use some type of MD5 / random seed thing)... if you distribute 10 bags of w00t incrementally, you just have to make sure that it gets distributed ~evenly~ across your population.

If IP's are bad then you can use cookies and exclude the use case where a non-cookied user gets offered a bag of w00t.

If you notice that a particular IP, cookie, or address range has an extreme amount of traffic, make the bag of w00t available to them proportionally later / last, so that occasional / steady / slow visitors are given opportunities before heavy / rapid / probable bot users.

--Robert

answered Feb 12, 2009 at 23:59

community wiki
Robert

0

I would recommend a firewall-based solution. Netfilter/iptables, as most firewalls, allows you to set a limit to the maximum number of new page requests per unit time.

For example, to limit the number of page views dispensed to something human -- say, 6 requests every 30 second -- you could issue the following rules:

```
iptables -N BADGUY
iptables -t filter -I BADGUY -m recent --set --
name badguys

iptables -A INPUT -p tcp --dport http -m state --
state NEW -m recent --name http --set
iptables -A INPUT -p tcp --dport http -m state --
state NEW -m recent --name http --rcheck --seconds
30 --hitcount 6 -j BADGUY
iptables -A INPUT -p tcp --dport http -m state --
state NEW -m recent --name http --rcheck --seconds
3 --hitcount 2 -j DROP
```

Note that this limit would apply to each visitor independently, so one user's misuse of the site wouldn't affect any other visitor.

Hope this helps!

Share Improve this answer

Follow

answered Feb 13, 2009 at 0:46

community wiki
v4lkyrius

0

You could reduce the load on your server by having the RSS and HTML update at the same time, so there's no incentive for the bots to screenscrape your site. Of course this gives the bots and advantage in buying your gear.

If you only accept payments via credit card (might be the case, might not be, but it shows my line of thinking) only allow a user to buy a BOC once every 10 sales with the same account and/or credit card. It's easy for a script kiddie to get a swarm of IPs, less easy for them to get a whole heap of credit cards together. And as you've said IPs are really hard to ban, while temporary bans on credit cards should be a walk in the park.

You could let everyone know what the limit is, or you could just tell them that because of the high demand and/or bot interest there's throttling implemented on the purchasing while being unspecific about the mechanism.

Each attempt to purchase during the throttling period could trigger an exponential backoff - you buy a BOC, you have to what for 10 sales to pass before you try again. You try again anyway on the next sale, and now you have to wait 20 sales, then 40, then 80...

This is only really useful if it's really unlikely that a human user would manage to get a BOC twice in less than 10 sales. Tune the number as appropriate.

Share Improve this answer answered Feb 13, 2009 at 0:52

Follow

community wiki
Dave

0

There are a few solutions you could take, based on the level of complexity you want to get into.

These are all based on IP tracking, which falls apart somewhat under botnets and cloud computing, but should thwart the vast majority of botters. The chances that Joe Random has a cloud of bots at his disposal is far lower than the chance that he's just running a Woot bot he downloaded somewhere so he can get his bag of crap.

Plain Old Throttling

At a very basic, crude level, you could throttle requests per IP per time period. Do some analysis and determine that a legitimate user will access the site no more than X times per hour. Cap requests per IP per hour at that number, and bots will have to drastically reduce their polling frequency, or they'll lock themselves out for the next 58 minutes and be completely blind. That doesn't address the bot problem by itself, but it does reduce load, and increases the chance that legitimate users will have a shot at the item.

Adaptive Throttling

An variant on that solution might be to implement a load balancing queue, where the number of requests that one has made recently counts against your position in the queue. That is, if you keep slamming the site, your requests become lower priority. In a high-traffic situation like the bag of crap sales, this would give legitimate users an advantage over the bots in that they would have a higher connection priority, and would be getting pages back more quickly, while the bots continue to wait and wait until traffic dies down enough that their number comes up.

End-of-the-line captcha

Third, while you don't want to bother with captchas, a captcha at the very end of the process, right before the transaction is completed, may not be a bad idea. At that point, people have committed to the sale, and are likely to go through with it even with the mild added annoyance. It prevents bots from completing the sale, which means that at a minimum all they can do is hammer your site to try to alert a human about the sale as quickly as possible. That doesn't solve the problem, but it does mean that the humans have a far, far better chance of obtaining sales than the bots do currently. It's not a solution, but it's an improvement.

A combination of the above

Implement basic, generous throttling to stop the most abusive of bots, while taking into account the potential for multiple legitimate users behind a single corporate IP. The

cutoff number would be very high - you cited bots hitting your site 10x/sec, which is 2.16 million requests/hour, which is obviously *far* above any legitimate usage, even for the largest corporate networks or shared IPs.

Implement the load balancing queue so that you're penalized for taking up more than your share of server connections and bandwidth. This penalizes people in the shared corporate pools, but it doesn't prevent them from using the site, and their violation should be far less terrible than your botters, so their penalization should be less severe.

Finally, if you have exceeded some threshold for requests-per-hour (which may be far, far, far lower than the "automatically drop the connection" cutoff), then require that the user validate with a captcha.

That way, the users who are legitimately using the site and only have 84 requests per hour, even when they're mega-excited, don't notice a change in the site's slow at all. However, Joe Botter finds himself stuck with a dilemma. He can either:

- Blow out his request quota with his current behavior and not be able to access the site at all, **or**
- Request just enough to not blow the request quota, which gives him realtime information at lower traffic levels, but causes him to have massive delays between requests during high-traffic times, which

severely compromises his ability to complete a sale before inventory is exhausted, **or**

- Request more than the average user and end up getting stuck behind a captcha, **or**
- Request no more than the average user, and thus have no advantage over the average user.

Only the abusive users suffer degradation of service, or an increase in complexity. Legitimate users won't notice a single change, except that they have an easier time buying their bags of crap.

Addendum

Throttle requests for unregistered users at rates far below registered users. That way, a bot owner would have to be running a bot via an authenticated account to get past what should be a relatively restrictive throttling rate.

The inventive botters will then register multiple user IDs and use those to achieve their desired query rate; you can combat that by considering any IDs that show from the same IP in a given period to be the same ID, and subject to shared throttling.

That leaves the botter with no recourse but to run a network of bots, with one bot per IP, and a registered Woot account per bot. This is, unfortunately, effectively indistinguishable from a large number of unassociated legitimate users.

You could use this strategy in conjunction with one or more of the above strategies with the goal to produce the overall effect of providing the best service to registered users who do not engage in abusive usage patterns, while progressively penalizing other users, both registered and unregistered, according to their status (anon or registered) and the level of abuse as determined by your traffic metrics.

Share Improve this answer Follow

edited Feb 13, 2009 at 1:55

community wiki
3 revs
cheald

0

my first thought was that you say the bots are scraping your webpage, which would suggest they are only picking up the HTML content. So having your order screen verify (from the http-logs) that an offer-related graphic was loaded from the bot

Share Improve this answer Follow

answered Feb 13, 2009 at 3:30

community wiki
bot

0

Develop a front page component and shopping cart that do not run natively in the brower. If you use something like Flex/Flash or Silverlight, it is much more difficult to scrape, and you have full control over the server communication, and thus can shield the content completely from scripters.

Share Improve this answer Follow

answered Feb 13, 2009 at 3:44

community wiki
cdonner

0

This only needs to be a problem if the bot users are paying with invalid credit cards or something. So how about a non-technical solution?

Treat the bot users as normal users as long as their payments are valid and make sure you have enough in stock to satisfy the total demand.

Result: more sales. You're in business to make money, right?

Share Improve this answer Follow

answered Feb 13, 2009 at 4:24

community wiki
Seun Osewa

0

To guarantee selling items only to non-scripted humans, could you detect inhumanly quick responses between the item being displayed on the front page and an order being made? This turns the delay tactic on its head, instead of handicapping everyone artificially through a .5 second delay, allow requests as fast as possible and smack bots that are clearly superhuman:)

There is some physical limit to how fast a user can click and make a decision, and by detecting *after* all the requests have gone through (as opposed to purposely slowing down all interacts), you don't effect performance of non-scripted humans.

If only using CAPTCHAs *some* of the time is acceptable, you could increase the delay time to fast-human (as opposed to superhuman) and require a post confirmation CAPTCHA if someone clicks really fast. Akin to how some sites require CAPTCHA confirmation if someone posts multiple posts quickly.

Sadly I don't know of any good ways to stop screen scrapers of your product listings :(

Share Improve this answer Follow

answered Feb 13, 2009 at 5:25

community wiki
Ryan

0

I'm just wondering if there might be a simple solution to this.

I assume that the message indicating the crap sale is posted in text and this is the bit of information the scrapers look for.

What if you made the announcement using an image instead? Doing so might pose some design problems but they could be overcome and possibly serve as the impetus for some ingenious creativity.

Issue #1
There would have to be some design space dedicated to

an image. (Want to be really tricky? Rotate a local ad through this slot. Of course the image's name would need to be static to avoid giving scrapers a scent. That's one slot that would never have to worry about ad-blindness...)

Issue #2
RSS. I'm not sure if everyone can view images in their feed readers. If enough of your users can, then you could start sending a daily feed update consisting of an image. You could send whatever miscellaneous stuff you wanted on most days and then switch it for your crap sale alert as desired.

I don't know... would they just program their bots to hit your site every time a feed item went out?

Other issues? Probably a lot. Maybe this will help with some brainstorming, though.

Take care,
Brian

Share Improve this answer Follow

answered Feb 13, 2009 at 6:05

community wiki

0

Here are some valid assumptions for you to make:

- Any automated solution can and will be broken.

- Making the site completely require human input (eg CAPTCHA) will greatly increase the difficulty of logging in/checking out/etc.
- You have a limited number of Bandoliers of Cabbage to sell.
- You can track users by session via a client-side cookie.
- You aren't dealing with extremely hardcore criminals here; these are simply technical people who are bending, but not breaking, the law. Successful orders via bots will go to the person's home, and likely not some third-party mail drop.

The solution isn't a technical one. It's a policy one.

- Log all client session ids on your webserver.
- Enact a "limited bots" policy; say, one screen scrape every X seconds, to give people with regular browsers the ability to hit refresh. Any user found to be going over this limit doesn't win the woot.
- Follow this up by sending known bot owners a bunch of Leakfrogs.

answered Feb 13, 2009 at 6:07

community wiki
Mark

Here is what I'd do:

1. Require all bidders for bag of crap sales to register with the site.
2. When you want to start a sale, post "BOC sale starting soon, check your email to see if you are eligible" on your main page.
3. Send out invitations to a random selection of the registered players, with a url unique to that particular sale when sale starts.
4. Ensure the URL used is different for each sales event.
5. Tweak the random selection invitation algorithm to pull down elibiblity for frequent winners, based upon Credit Card used for purchase, paypal account, or shipping address.

This thwarts the bots, as your main page only shows the pending BOC event. The bots will not have access to the URL without recieving it in email, and have no guarantee they will recieve it at all.

If you are concerned about sales impact, you could also incentivize participation by giving away one or two BOC's for each sale. If you don't see enough uptake on an offer in a given time interval, you automatically mail additional registered users, increasing the participant pool in each offer.

Viola. Level playing field, without tons of heuristics and web traffic analysis. System can still be gamed by people setting up huge numbers of email accounts, but tweaking participant selection criteria by CC#, paypal account, shipping address mitigates this.

Share Improve this answer Follow

answered Feb 13, 2009 at 6:17

community wiki
Simon

0

What about the NoBot Control from the ASP.net AJAX control toolkit?

It does some automated javascript request and timing tricks to prevent bots from accessing the site with NO user interaction.

Sorry if this doesn't meet some requirement, i'll just have to call
tl;dr >D

Share Improve this answer Follow

answered Feb 13, 2009 at 6:59

community wiki
TJB

0

Turn certain parts of the page into images so the bots can't understand them.

For example create small images of the integers 0-9, the dollar sign, and the decimal point. Cache the images on the client's computer when the page loads... then display the price using images chosen via code running server-side. Most human users won't notice the difference and the bots won't know the prices of any items.

Share Improve this answer Follow

answered Feb 13, 2009 at 7:26

community wiki
MrDatabase

Prev 1 2 3 4 5 Next