How to enable DDoS protection?

Asked 11 years, 11 months ago Modified 6 years, 3 months ago

Viewed 70k times Part of PHP Collective

93

DDoS (Distributed Denial of Service Attacks) are generally blocked on a server level right?

Is there a way to block it on a PHP level, or at least reduce it?

If not, what is the fastest and most common way to stop DDoS attacks?

PHP php security ddos denial-of-service

Share

Improve this question

Follow

edited Jun 10, 2013 at 18:13

Prix

19.5k ● 16 ● 76 ● 135

asked Jan 23, 2013 at 10:54

rockstardev

13.5k ● 40 ● 170 ● 302

From having nothing better, here there is a list of Apache modules that might potentially help. It seems not very rich, however, and the two projects of four (as searched by 'dos') appear pointing to nowhere. – Audrius Meškauskas Jan 30, 2013 at 10:21

Something like this? mod-antiloris.sourceforge.net – Kerem Jan 30, 2013 at 11:40

A DDOS is successful when the system can no longer process all the request the attack is throwing at it. If you add code in your application that checks each request for "Is this a DDOS attack" then that code is actually doing more harm then good, because it just takes a little bit more resources for each request. This would only work in a very restrictive environment where anonymous request arent allowed anyway. – Hugo Delsing Feb 4, 2013 at 13:49

10 Answers

Sorted by: Highest score (default)

198

✓

+50

DDOS is a family of attacks which overwhelm key systems in the datacenter including:

- The hosting center's network connection to the internet
- The hosting center's internal network and routers
- Your firewall and load balancers
- Your web servers, application servers and database.

Before you start on building your DDOS defence, consider what the worst-case value-at-risk is. For a non-critical, free-to-use service for a small community, the total value at risk might be peanuts. For a paid-for, public-facing, mission-critical system for an established multi-billion dollar business, the value might be the worth of the company. In this latter case, you shouldn't be using

StackExchange :) Anyway, to defend against DDOS, you need a defence in-depth approach:

1. **Work with your hosting center** to understand the services they offer, including IP and port filtering at their network connections to the internet and firewall services they offer. This is critical: Many sites are pulled from the internet *by the hosting company* as the hosting company deals with the data center-wide disruption caused by the DDOS to one customer. Also, during an DDOS attack, you will be working very closely with the hosting center's staff, so know their emergency numbers and be on good terms with them :) They should be able to block of whole international regions, completely block specific services or network protocols and other broad-spectrum defensive measures, or alternatively allow only whitelisted IPs (depending on your business model)

2. While on the hosting center - use a **Content Delivery Network** to distribute (mainly static) services close to your end users and hide your real servers from the DDOS architects. The full CDN is too big for a DDOS to take out all nodes in all countries; if the DDOS is focused on one country, at least other users are still OK.

3. Keep all your systems and software packages **updated with the latest security patches** - and I mean all of them:

- Managed switches - yup these sometimes need updating
- Routers
- Firewalls
- Load balancers
- Operating systems
- Web servers
- Languages and their libraries

4. Ensure that you have a **good firewall or security appliance** set up *and regularly reviewed by a qualified security expert*. Strong rules on the firewall are a good defence against many simple attacks. It's also useful to be able to manage bandwidth available for each open service.
5. Have good **network monitoring tools** in place - this can help you understand:
    - That you're under attack rather than simply being under heavy load
    - Where the attack is coming from (which may include countries you don't normally do business with) and
    - What the attack actually is (ports, services, protocols, IPs and packet contents)
6. The attack might simply be heavy use of legitimate web site services (eg hitting 'legal' URIs running queries or inserting/updating/deleting data) -

thousands or millions of requests coming from tens to millions of different IP addresses will bring a site to its knees. Alternatively, some services might be so expensive to run that only a few requests cause a DOS - think a really expensive report. So you need good **application level monitoring** of what is going on:

- Which services have been invoked and what arguments/data are sent (i.e. logging in your application)
- Which users are doing the invoking and from which IPs (i.e. logging in your application)
- What queries and inserts/updates/deletes the DB is performing
- Load average, CPU utilization, disk i/o, network traffic on all computers (and VMs) in your system
- Making sure that all this information is easily retrievable and that you can correlate logs from different computers and services (i.e. ensure all computers are time synchronized using ntp).

7. **Sensible constraints and limits in your application**. For example, you might:

   - Use a QoS feature in the load balancer to send all anonymous sessions to separate application servers in your cluster, while logged-on users use another set. This prevents an application-level anonymous DDOS taking out valuable customers

    - Using a strong CAPCHA to protect anonymous services
    - Session timeouts
    - Have a session-limit or rate-limit on certain types of request like reports. Ensure that you can turn off anonymous access if necessary
    - Ensure that a user has a limit to the number of concurrent sessions (to prevent a hacked account logging on a million times)
    - Have different database application users for different services (eg transactional use vs. reporting use) and use database resource management to prevent one type of web request from overwhelming all others
    - If possible make these constraints dynamic, or at least configurable. This way, while you are under attack, you can set aggressive temporary limits in place ('throttling' the attack), such as only one session per user, and no anonymous access. This is certainly not great for your customers, but a lot better than having no service at all.

8. Last, but not least, write a **DOS Response Plan** document and get this internally reviewed by all relevant parties: Business, Management, the SW dev team, the IT team and a security expert. The process

> of writing the document will cause you and your team to think through the issues and help you to be prepared if the worst should happen at 3am on your day off. The document should cover (among other things):
>
> - What is at risk, and the cost to the business
> - Measures taken to protect the assets
> - How an attack is detected
> - The planned response and escalation procedure
> - Processes to keep the system and this document up-to-date

So, preamble aside, here are some specific answers:

> DDOS are generally blocked on a server level, right?

Not really - most of the worst DDOS attacks are low-level (at the IP packet level) and are handled by routing rules, firewalls, and security devices developed to handle DDOS attacks.

> Is there a way to block it on a PHP level, or at least reduce it?

Some DDOS attacks are aimed at the application itself, sending valid URIs and HTTP requests. When the rate of requests goes up, your server(s) begin to struggle and you will have an SLA outage. In this case, there are things you can do at the PHP level:

- Application level monitoring: Ensure each service/page logs requests in a way that you can see what is going on (so you can take actions to mitigate the attack). Some ideas:
    - Have a log format that you can easily load into a log tool (or Excel or similar), and parse with command-line tools (grep, sed, awk). Remember that a DDOS will generate millions of lines of log. You will likely need to slice'n'dice your logs (especially with respect to URI, time, IP and user) to work out what is going on, and need to generate data such as:
        - What URIs are being accessed
        - What URIs are failing at a high rate (a likely indicator of the specific URIs the attackers are attacking)
        - Which users are accessing the service
        - How many IPs are each user accessing the service from
        - What URIs are anonymous users accessing
        - What arguments are being used for a given service
        - Audit a specific users actions
    - Log the IP address of each request. DON'T reverse DNS this - ironically the cost of doing

    this makes a DDOS easier for the attackers
    - Log the whole URI and HTTP method, eg "GET http://example.com/path/to/service?arg1=ddos"
    - Log the User ID if present
    - Log important HTTP arguments
- Sensible rate limits: You might implement limits on how many requests a given IP or User can make in a given time period. Could a legitimate customer make more than 10 requests per second? Can anonymous users access expensive reports at all?
- CAPTCHA for anonymous access: Implement a CAPTCHA for all anonymous requests to verify that the user is a person, not a DDOS bot.

> What's the fastest and most common way to stop DDOS attacks?

The fastest is probably to give in to the blackmail, although this might not be desirable.

Otherwise, the first thing you to do is contact your hosting and/or CDN provider and work with them (if they haven't contacted you already asking what the hell is going on...). When a DDOS occurs, it will likely collaterally affect other customers of the hosting provider, and the provider may be under considerable pressure to shut down your site simply to protect their resources. Be prepared to share your logs (any and all information) with the provider; these logs, combined with their network monitors, may together provide enough information to block/mitigate the attack.

If you are expecting a DDOS, it's a very good idea to qualify your hosting provider on the level of protection they can provide. They should have DDOS experience and tools to mitigate it - understand their tools, processes and escalation procedures. Also ask about what support the hosting provider has from *their* upstream providers. These services might mean more up-front or monthly cost, but treat this as an insurance policy.

While under attack, you will need to grab your logs and mine them - try and work out the pattern of the attack. You should consider switching off anonymous access and throttling the services under attack (i.e. decrease the application's rate limit for the service).

If lucky and you have a small, fixed customer-base, you might be able to determine your valid customers IP addresses. If this is the case, you might switch to a white-list approach for a short while. Make sure all your customers know this is going on so they can call if they need to access from a new IP :)

Doug McClean has some great advice at: https://stackoverflow.com/a/1029613/1395668

Share Improve this answer Follow

edited May 23, 2017 at 12:02

Community Bot

1 ● 1

answered Jan 30, 2013 at 8:00

Andrew Alcock
19.6k ● 4 ● 43 ● 60

23

According the PHP part of the question;

Although I don't rely on PHP for this, it could be implemented but needs to consider all these possiblities or more;

1. Attacker may change IP for each request
2. Attacker may pass parameter(s) to URI that target site doesn't care these parameter(s)
3. Attacker may restart the session before expiry ...

Simple pseudo;

```
<?php
// Assuming session is already started
$uri = md5($_SERVER['REQUEST_URI']);
$exp = 3; // 3 seconds
$hash = $uri .'|'. time();
if (!isset($_SESSION['ddos'])) {
    $_SESSION['ddos'] = $hash;
}

list($_uri, $_exp) = explode('|', $_SESSION['ddos']);
if ($_uri == $uri && time() - $_exp < $exp) {
    header('HTTP/1.1 503 Service Unavailable');
    // die('Easy!');
    die;
}

// Save last request
```

```
$_SESSION['ddos'] = $hash;
?>
```

Share Improve this answer Follow

edited Feb 4, 2013 at 9:14

answered Jan 30, 2013 at 14:41

Kerem
11.5k ● 5 ● 60 ● 58

9

The php level is too late in the request chain.

Putting your apache server behind an open source appliance may be a good option for you.

http://tengine.taobao.org/ has some documentation and source code more modules aimed at DDOS prevention. It is a expansion of nginx, so you can easily set it up as a reverse proxy for your apache instance.

See: http://blog.zhuzhaoyuan.com/2012/01/a-mechanism-to-help-write-web-application-firewalls-for-nginx/ for how to fight collision has DoS attacks.

Totally forgot too, http://www.cloudflare.com is one the top free web application firewall, they have free and paid plans and will save your ass from DDOS we use it for alot of our high traffic sites just for its caching capabilities. It is awsome!

Share Improve this answer

edited Feb 5, 2013 at 17:49

Follow

answered Feb 4, 2013 at 23:23

j_mcnally
6,958 ● 2 ● 33 ● 46

6

You can not do this in PHP level. DDOS is a kind of attack that send too many requests to your webserver. Your webserver will reject request before it call your PHP script.

If you are using Apache, here is some tips from Apache: http://httpd.apache.org/docs/trunk/misc/security_tips.html

Share Improve this answer

Follow

answered Jan 29, 2013 at 5:52

ndlinh
1,365 ● 9 ● 14

5

DDoS is best handled by very expensive, purpose-built network appliances. Hosts are generally not good at doing DDoS protection because they are subject to relatively low performance, state exhaustion, limited bandwidth, etc. Use of iptables, apache mods, and similar services can help in some situations if you have no access to DDoS mitigation hardware or a DDoS mitigation service, but it is far from ideal and still leaves you at risk of attack.

Share Improve this answer

answered Jan 29, 2013 at 16:30

Follow

ryan
75 ● 1

9 We need a solution, and if there is no an ideal solution, were need partial solutions. Nobody needs an advice of kind "you just cannot do this". – Audrius Meškauskas Jan 30, 2013 at 9:51

"very expensive, purpose-built network appliances"? No, that's simply not true – Nico Haase May 5, 2021 at 15:14

5

Do **NOT** use PHP-based protection, it's horrible and will hardly have an impact at all! Configure your webserver to rate-limit requests, for example in Nginx using the limit_req module (http://nginx.org/en/docs/http/ngx_http_limit_req_module.html)

Although, I would recommend using CloudFlare to combat layer-4 - however not layer-7 based attacks unless you're willing to pay.

Share Improve this answer Follow

answered Apr 24, 2015 at 23:17
JustLloyd
129 ● 3 ● 8

yes, worked great for me as well, my experience here softwareengineeringsolutions.co.uk/… – E Ciotti Aug 13, 2018 at 20:48

4

There are plugins you can use in apache for ddos/dos. Good start here http://www.debianadmin.com/how-to-protect-apache-against-dosddos-or-brute-force-attacks.html

If you're on LEMP, you can check here. http://nginx.org/en/docs/http/ngx_http_limit_conn_module.html

These are good inexpensive starting points.

Share Improve this answer Follow

answered Feb 5, 2013 at 0:47

JasonG
5,962 ● 4 ● 40 ● 69

4

How about something like this on PHP side:

```
//if user does not change IP, then ban the IP when mor
second are detected in 1 second
$limitps = 10;
if (!isset($_SESSION['first_request'])){
    $_SESSION['requests'] = 0;
    $_SESSION['first_request'] = $_SERVER['REQUEST_TIM
}
$_SESSION['requests']++;
if ($_SESSION['requests']>=10 && strtotime($_SERVER['R
strtotime($_SESSION['first_request'])<=1){
    //write the IP to a banned_ips.log file and config
retrieve the banned ips from there - now you will be h
of PHP
    $_SESSION['banip']==1;
}elseif(strtotime($_SERVER['REQUEST_TIME'])-
strtotime($_SESSION['first_request']) > 2){
    $_SESSION['requests'] = 0;
    $_SESSION['first_request'] = $_SERVER['REQUEST_TIM
}
```

```
if ($_SESSION['banip']==1) {
    header('HTTP/1.1 503 Service Unavailable');
    die;
}
```

Share Improve this answer Follow

answered Feb 19, 2015 at 11:24

NVG
3,313 ● 10 ● 42 ● 62

1 There are some errors in this code: 10 is hard coded, strtotime is not needed in the server and session variables, and $_SESSION['ban_up'] == 1 should be = 1. – Jason Silver Jul 27, 2018 at 18:32

"How about" is a good question, as this doesn't even try to resolve DDoS attacks – Nico Haase May 5, 2021 at 15:12

4

DDOS are generally blocked on a server level, Please enable DDOS protection in your Server Level. Please check the below notes for DDOS protections.

Apache HTTP Server configuration settings that can help prevent DDOS problems:

The RequestReadTimeout directive allows to limit the time a client may take to send the request.

Allow 10 seconds to receive the request including the headers and 30 seconds for receiving the request body:

```
RequestReadTimeout header=10 body=30
```

Allow at least 10 seconds to receive the request body. If the client sends data, increase the timeout by 1 second for every 1000 bytes received, with no upper limit for the timeout (except for the limit given indirectly by LimitRequestBody):

```
RequestReadTimeout body=10,MinRate=1000

RequestReadTimeout header=10-30,MinRate=500
RequestReadTimeout header=20-40,MinRate=500 body=20,Mi
```

The KeepAliveTimeout directive may be also lowered on sites that are subject to DoS attacks. Some sites even turn off the keepalives completely via KeepAlive, which has of course other drawbacks on performance. The values of various timeout-related directives provided by other modules should be checked.

The directives LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine, and LimitXMLRequestBody should be carefully configured to limit resource consumption triggered by client input. Tune the MaxRequestWorkers directive to allow the server to handle the maximum number of simultaneous connections without running out of resources.

Share Improve this answer Follow

answered Aug 15, 2017 at 15:21

Raja Ram T
9,001 ● 4 ● 33 ● 31

Anti DDOS steps:

3

- The very first important thing is to identify the ddos attack first. Identifying the ddos attack more early means more better for your server .
- Getting better bandwidth available for your server. Always keep more than enough bandwidth which is required to for your server. This won't prevent DDOS attack but it will take longer time. By which you will get some extra time to act.
- If you own your own web server then you can defend at network parameter by rate limit your router, add filters to drop packets to different sources of attacks, time out half opened connections more aggressively. Also set lower SYN, ICMP and UDP flood drop thresholds.
- If you don't have much idea about these things, then go and contact your hosting providers quickly. They can try their best prevent the DDOS attacks.
- There are also Special DDOS mitigation service provided by **Cloudflare** and many other companies. By which they can help you to prevent the DDOS attacks. Also many companies offer cheap **ddos protection** and **dos protection**.

Share Improve this answer Follow

edited Sep 5, 2018 at 8:13

galoget
724 ● 9 ● 15

answered Nov 29, 2016 at 17:09

IamNOOB
121 ● 1 ● 3 ● 11