Cloudflare Bot Management: machine learning and more

2020年5月6日 下午7:00

Alex Bocharov

Introduction

Building Cloudflare Bot Management platform is an exhilarating experience. It blends Distributed Systems, Web Development, Machine Learning, Security and Research (and every discipline in between) while fighting ever-adaptive and motivated adversaries at the same time.

This is the ongoing story of Bot Management at Cloudflare and also an introduction to a series of blog posts about the detection mechanisms powering it. I'll start with several definitions from the Bot Management world,

then introduce the product and technical requirements, leading to an overview of the platform we've built. Finally, I'll share details about the detection mechanisms powering our platform.

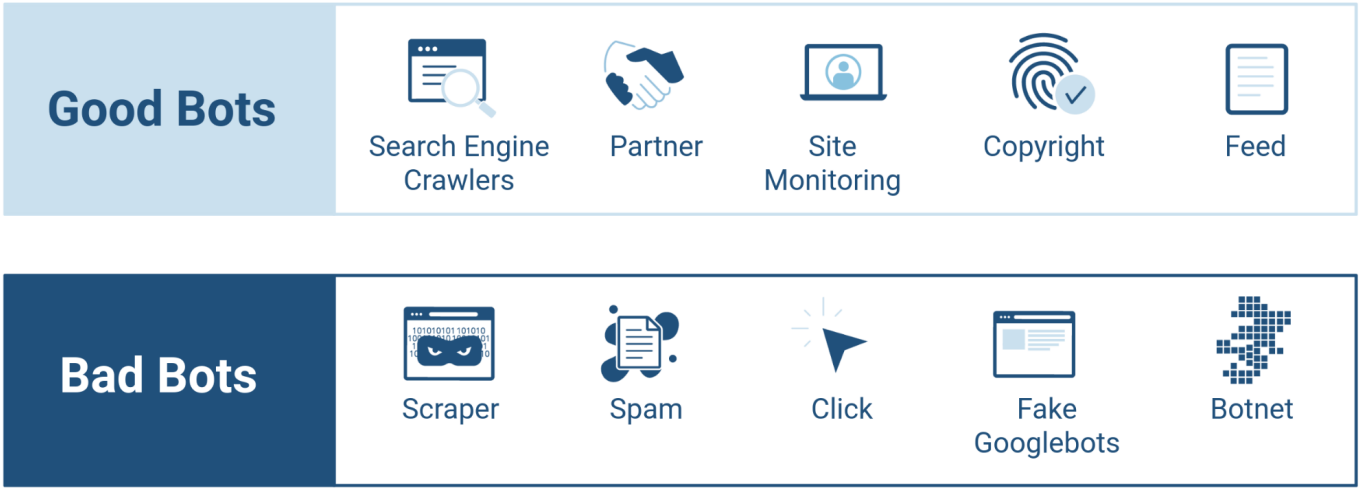
Let's start with Bot Management's nomenclature.

Some Definitions

Bot - an autonomous program on a network that can interact with computer systems or users, imitating or replacing a human user's behavior, performing repetitive tasks much faster than human users could.

Good bots - bots which are useful to businesses they interact with, e.g. search engine bots like Googlebot, Bingbot or bots that operate on social media platforms like Facebook Bot.

Bad bots - bots which are designed to perform malicious actions, ultimately hurting businesses, e.g. credential stuffing bots, third-party scraping bots, spam bots and sneakerbots.



Bot Management - blocking undesired or malicious Internet bot traffic while still allowing useful bots to access web properties by detecting bot activity, discerning between desirable and undesirable bot behavior, and identifying the sources of the undesirable activity.

WAF - a security system that monitors and controls network traffic based on a set of security rules.

Gathering requirements

Cloudflare has been stopping malicious bots from accessing websites or misusing APIs from the very beginning, at the same time helping the climate by offsetting the carbon costs from the bots. Over time it became clear that we needed a dedicated platform which would unite different bot fighting techniques and streamline the customer experience. In designing this new platform, we tried to fulfill the following key requirements.

- **Complete, not complex** - customers can turn on/off Bot Management with a single click of a button, to protect their websites, mobile applications, or APIs.

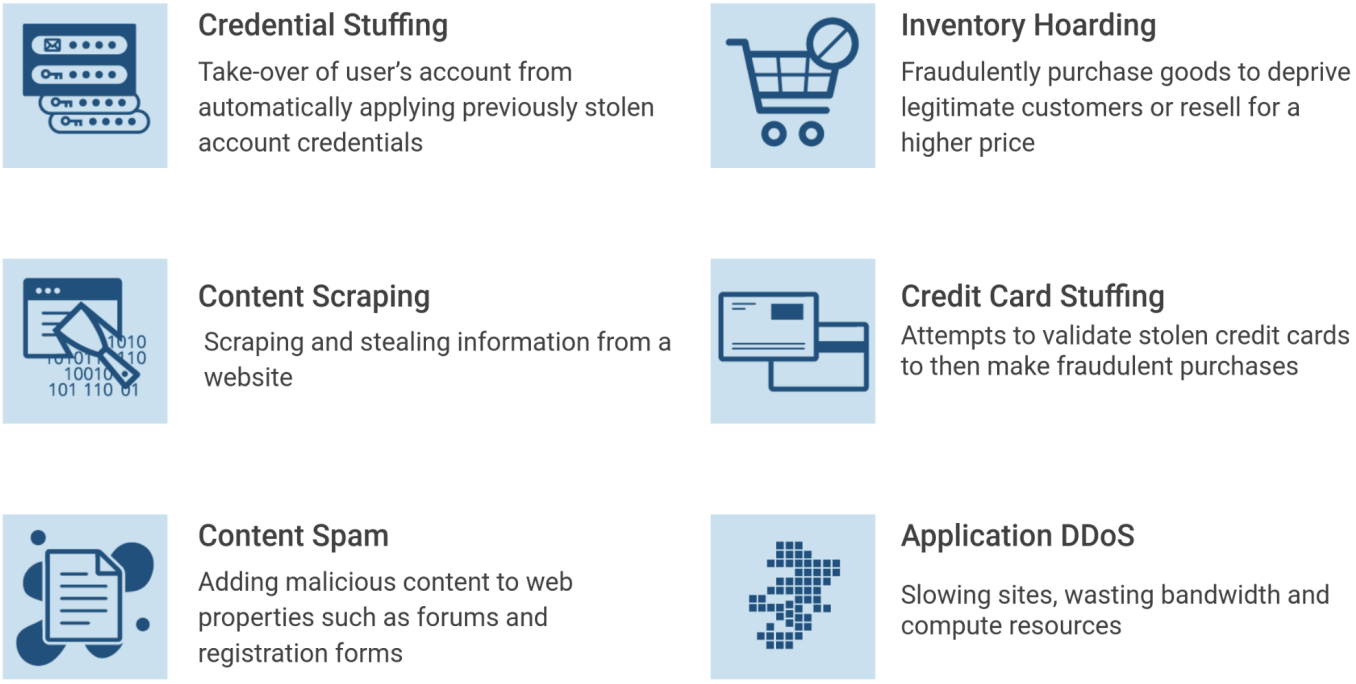
- **Trustworthy** - customers want to know whether they can trust the website visitor is who they say they are and provide a certainty indicator for that trust level.

- **Flexible** - customers should be able to define what subset of the traffic Bot Management mitigations should be applied to, e.g. only login URLs, pricing pages or sitewide.

- **Accurate** - Bot Management detections should have a very small error, e.g. none or very few human visitors ever should be mistakenly identified as bots.

- **Recoverable** - in case a wrong prediction was made, human visitors still should be able to access websites as well as good bots being let through.

Moreover, the goal for new Bot Management product was to make it work well on the following use cases:



Technical requirements

Additionally to the product requirements above, we engineers had a list of must-haves for the new Bot Management platform. The most critical were:

- **Scalability** - the platform should be able to calculate a score on every request, even at over 10 million requests per second.

- **Low latency** - detections must be performed extremely quickly, not slowing down request processing by more than 100 microseconds, and not requiring additional hardware.

- **Configurability** - it should be possible to configure what detections are applied on what traffic, including on per domain/data center/server level.

- **Modifiability** - the platform should be easily extensible with more detection mechanisms, different mitigation actions, richer analytics and logs.

- **Security** - no sensitive information from one customer should be used to build models that protect another customer.

- **Explainability & debuggability** - we should be able to explain and tune predictions in an intuitive way.

Equipped with these requirements, back in 2018, our small team of engineers got to work to design and build the next generation of Cloudflare Bot Management.

Meet the Score

“Simplicity is the ultimate sophistication.”
- Leonardo Da Vinci

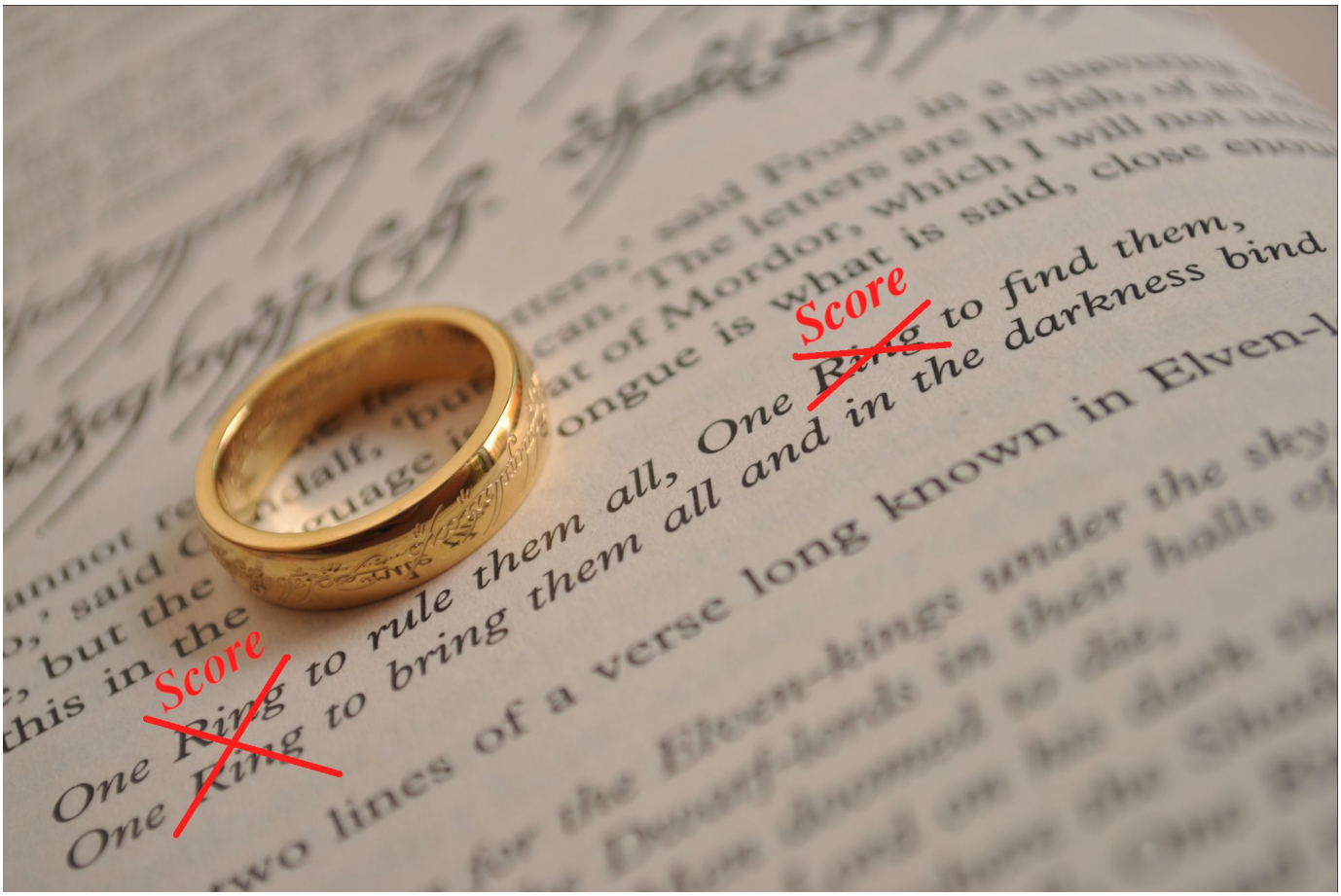
Cloudflare operates on a vast scale. At the time of this writing, this means covering 26M+ Internet properties, processing on average 11M requests per second (with peaks over 14M), and examining more than 250 request attributes from different protocol levels. The key question is how to harness the power of such “gargantuan” data to protect all of our customers from modern day cyberthreats in a simple, reliable and explainable way?

Bot management is hard. Some bots are much harder to detect and require looking at multiple dimensions of request attributes over a long time, and sometimes a single request attribute could give them away. More signals may help, but are they generalizable?

When we classify traffic, should customers decide what to do with it or are there decisions we can make on behalf of the customer? What concept could possibly address all these uncertainty problems and also help us to deliver on the requirements from above?

As you might’ve guessed from the section title, we came up with the concept of Trusted Score or simply **The Score** - one thing to rule them all - indicating

the likelihood between 0 and 100 whether a request originated from a human (high score) vs. an automated program (low score).



"One Ring to rule them all" by idreamlikecrazy, used under CC BY / Desaturated from original

Okay, let's imagine that we are able to assign such a score on every incoming HTTP/HTTPS request, what are we or the customer supposed to do with it? Maybe it's enough to provide such a score in the logs. Customers could then analyze them on their end, find the most frequent IPs with the lowest scores, and then use the Cloudflare Firewall to block those IPs. Although useful, such a process would be manual, prone to error and most importantly cannot be done in real time to protect the customer's Internet property.

Fortunately, around the same time we started worked on this system , our colleagues from the Firewall team had just announced Firewall Rules. This new capability provided customers the ability to control requests in a flexible

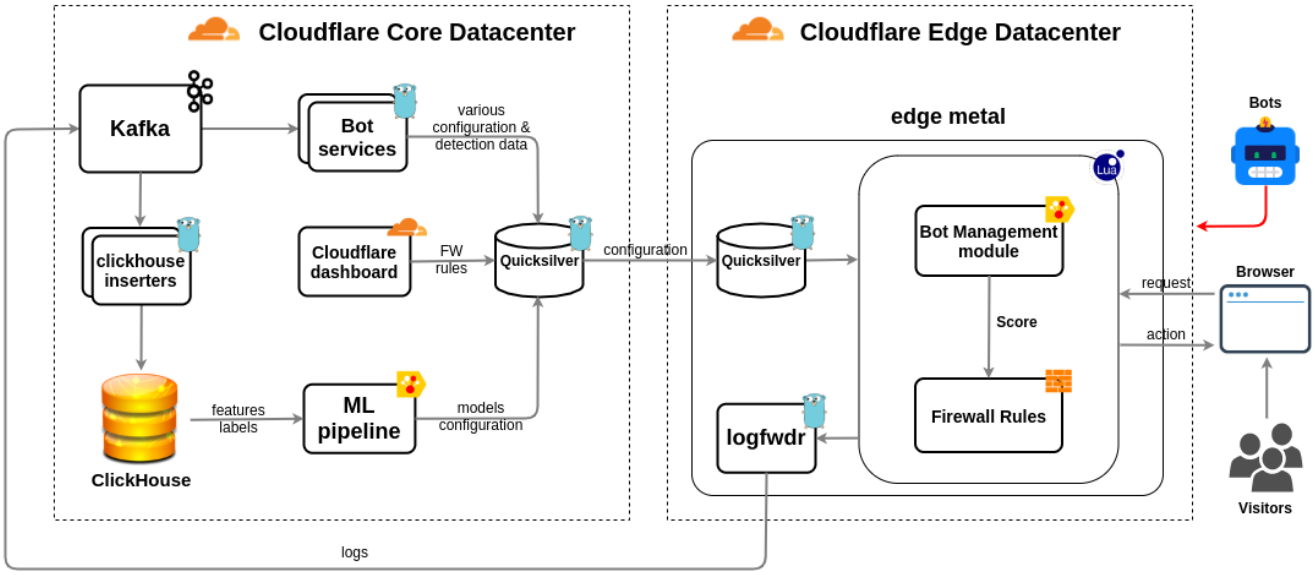
and intuitive way, inspired by the widely known Wireshark® language. Firewall rules supported a variety of request fields, and we thought - why not have the score be one of these fields? Customers could then write granular rules to block very specific attack types. That's how the `cf.bot_management.score` field was born.

Having a score in the heart of Cloudflare Bot Management addressed multiple product and technical requirements with one strike - it's simple, flexible, configurable, and it provides customers with telemetry about bots on a per request basis. Customers can adjust the score threshold in firewall rules, depending on their sensitivity to false positives/negatives. Additionally, this intuitive score allows us to extend our detection capabilities under the hood without customers needing to adjust any configuration.

So how can we produce this score and how hard is it? Let's explore it in the following section.

Architecture overview

What is powering the Bot Management score? The short answer is a set of microservices. Building this platform we tried to re-use as many pipelines, databases and components as we could, however many services had to be built from scratch. Let's have a look at overall architecture (this overly simplified version contains Bot Management related services):



Core Bot Management services

In a nutshell our systems process data received from the edge data centers, produce and store data required for bot detection mechanisms using the following technologies:

- **Databases & data stores** - Kafka, ClickHouse, Postgres, Redis, Ceph.
- **Programming languages** - Go, Rust, Python, Java, Bash.
- **Configuration & schema management** - Salt, Quicksilver, Cap'n Proto.
- **Containerization** - Docker, Kubernetes, Helm, Mesos/Marathon.

Each of these services is built with resilience, performance, observability and security in mind.

Edge Bot Management module

All bot detection mechanisms are applied on every request in real-time during the request processing stage in the Bot Management module running on

every machine at Cloudflare's edge locations. When a request comes in we extract and transform the required request attributes and feed them to our detection mechanisms. The Bot Management module produces the following output:

Firewall fields - Bot Management fields
- **cf.bot_management.score** - an integer indicating the likelihood between 0 and 100 whether a request originated from an automated program (low score) to a human (high score).
- **cf.bot_management.verified_bot** - a boolean indicating whether such request comes from a Cloudflare whitelisted bot.
- **cf.bot_management.static_resource** - a boolean indicating whether request matches file extensions for many types of static resources.

Cookies - most notably it produces **cf_bm**, which helps manage incoming traffic that matches criteria associated with bots.

JS challenges - for some of our detections and customers we inject into invisible JavaScript challenges, providing us with more signals for bot detection.

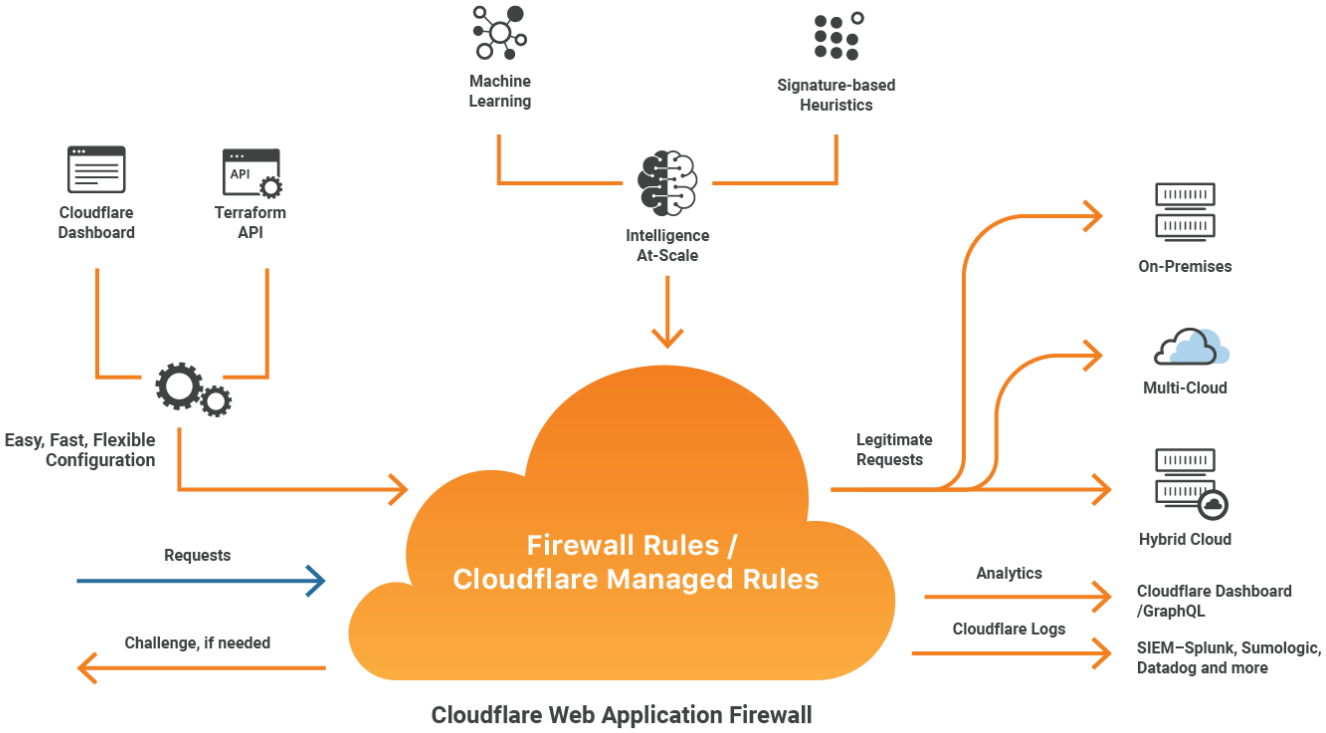
Detection logs - we log through our data pipelines to ClickHouse details about each applied detection, used features and flags, some of which are used for analytics and customer logs, while others are used to debug and improve our models.

Once the Bot Management module has produced the required fields, the Firewall takes over the actual bot mitigation.

Firewall integration

The Cloudflare Firewall's intuitive dashboard enables users to build powerful rules through easy clicks and also provides Terraform integration. Every request to the firewall is inspected against the rule engine. Suspicious requests can be blocked, challenged or logged as per the needs of the user

while legitimate requests are routed to the destination, based on the score produced by the Bot Management module and the configured threshold.



Firewall rules provide the following bot mitigation actions:

- **Log** - records matching requests in the Cloudflare Logs provided to customers.
- **Bypass** - allows customers to dynamically disable Cloudflare security features for a request.
- **Allow** - matching requests are exempt from challenge and block actions triggered by other Firewall Rules content.
- **Challenge (Captcha)** - useful for ensuring that the visitor accessing the site is human, and not automated.
- **JS Challenge** - useful for ensuring that bots and spam cannot access the requested resource; browsers, however, are free to satisfy the

challenge automatically.

- **Block** - matching requests are denied access to the site.

Our Firewall Analytics tool, powered by ClickHouse and GraphQL API, enables customers to quickly identify and investigate security threats using an intuitive interface. In addition to analytics, we provide detailed logs on all bots-related activity using either the Logpull API and/or LogPush, which provides the easy way to get your logs to your cloud storage.

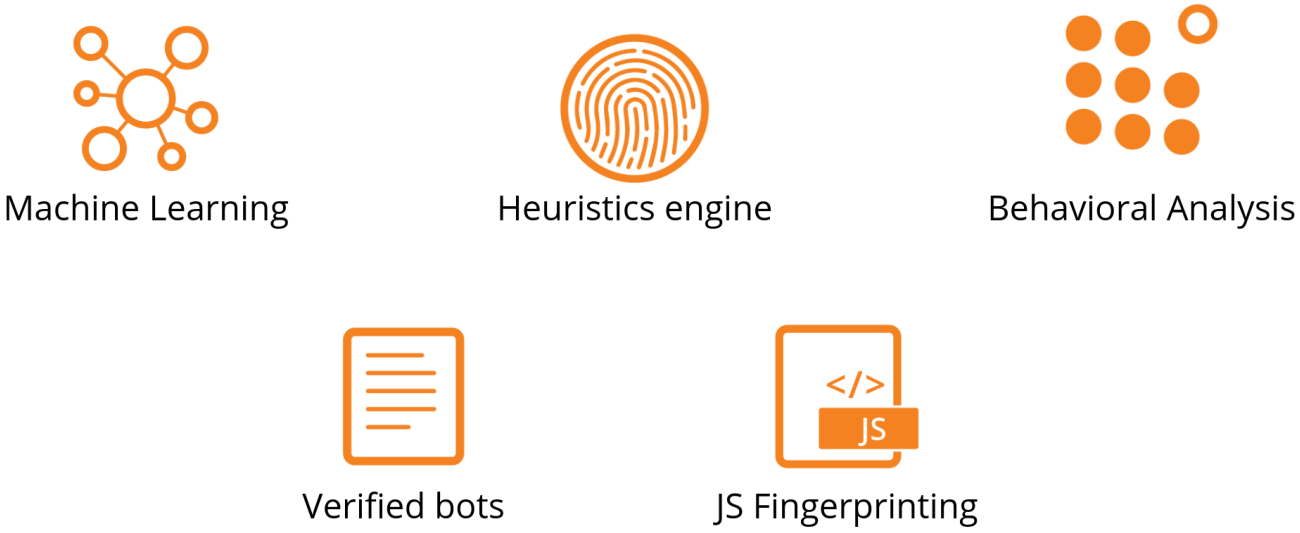
Cloudflare Workers integration

In case a customer wants more flexibility on what to do with the requests based on the score, e.g. they might want to inject new, or change existing, HTML page content, or serve incorrect data to the bots, or stall certain requests, Cloudflare Workers provide an option to do that. For example, using this small code-snippet, we can pass the score back to the origin server for more advanced real-time analysis or mitigation:

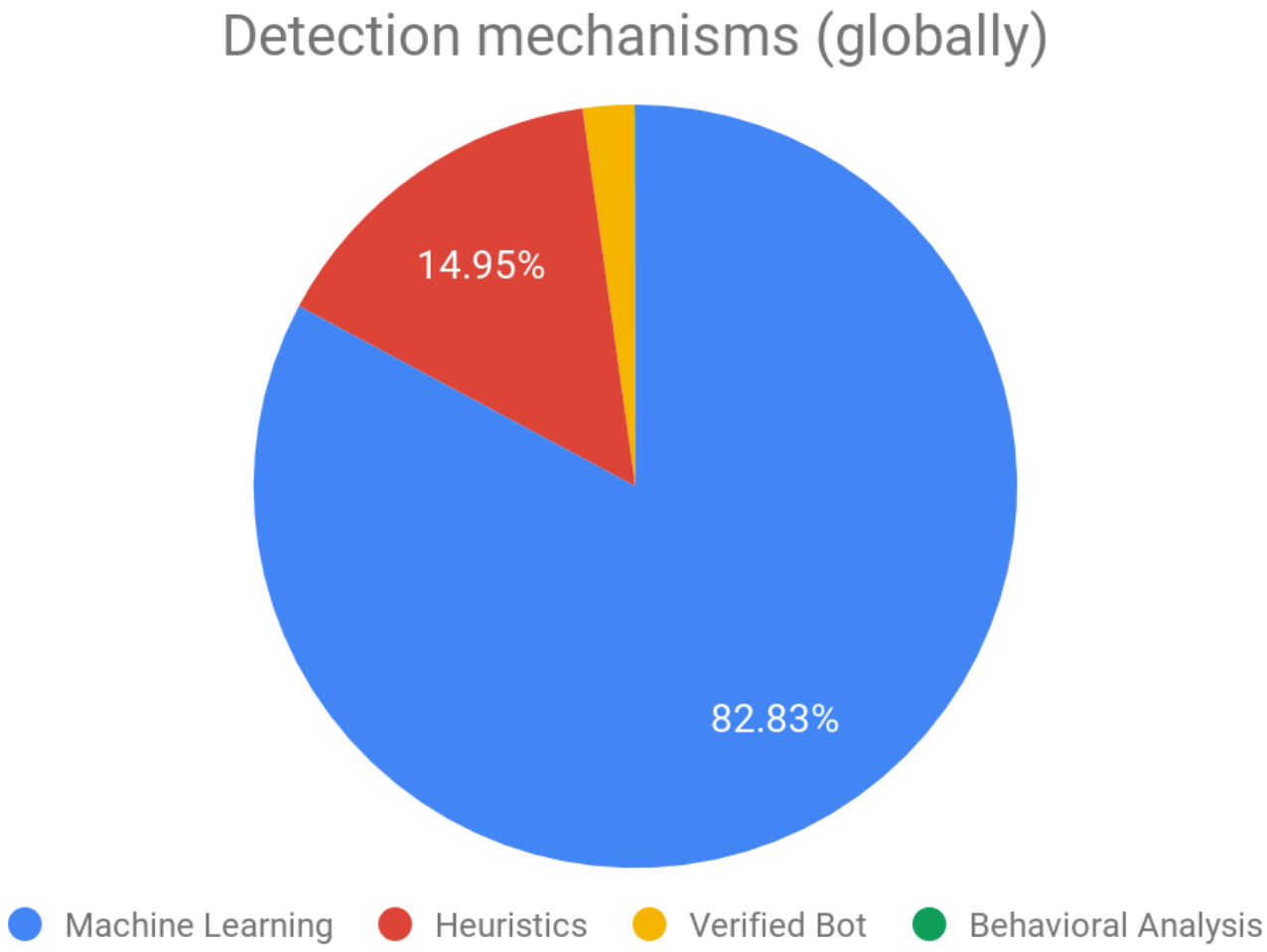
```
addEventListener('fetch', event => {
  event.respondWith(handleRequest(event.request))
})

async function handleRequest(request) {
  request = new Request(request);

  request.headers.set("Cf-Bot-Score", request.cf.bot_management.score)

  return fetch(request);
}
```

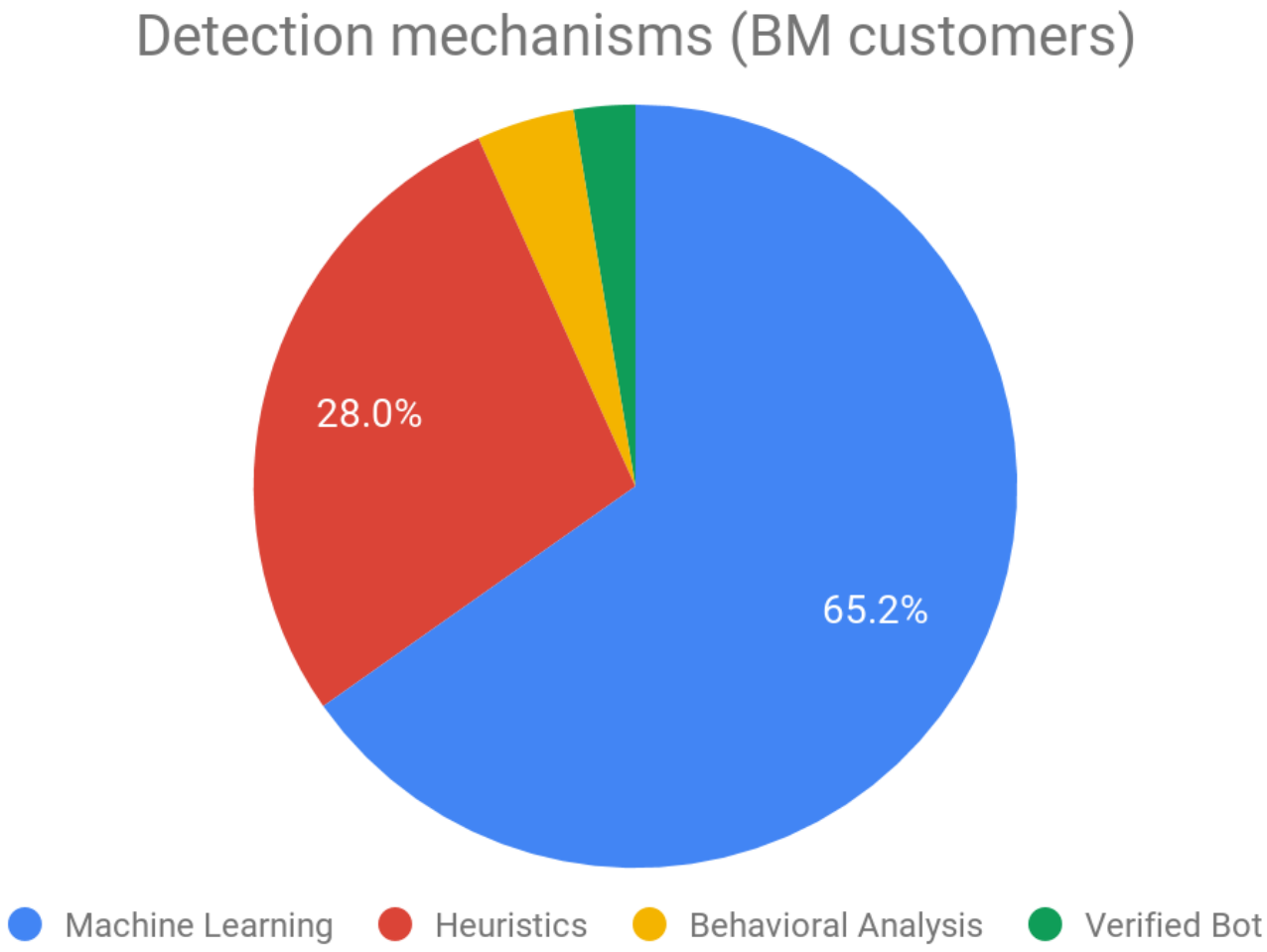
Now let's have a look into how a single score is produced using multiple detection mechanisms.

Detection mechanisms



The Cloudflare Bot Management platform currently uses five complementary detection mechanisms, producing their own scores, which we combine to form the single score going to the Firewall. Most of the detection mechanisms are applied on every request, while some are enabled on a per customer basis to better fit their needs.


Detection mechanisms (globally)
14.95%
82.83%
Machine Learning
Heuristics
Verified Bot
Behavioral Analysis



Having a score on every request for every customer has the following benefits:

- **Ease of onboarding** - even before we enable Bot Management in active mode, we're able to tell how well it's going to work for the specific customer, including providing historical trends about bot activity.

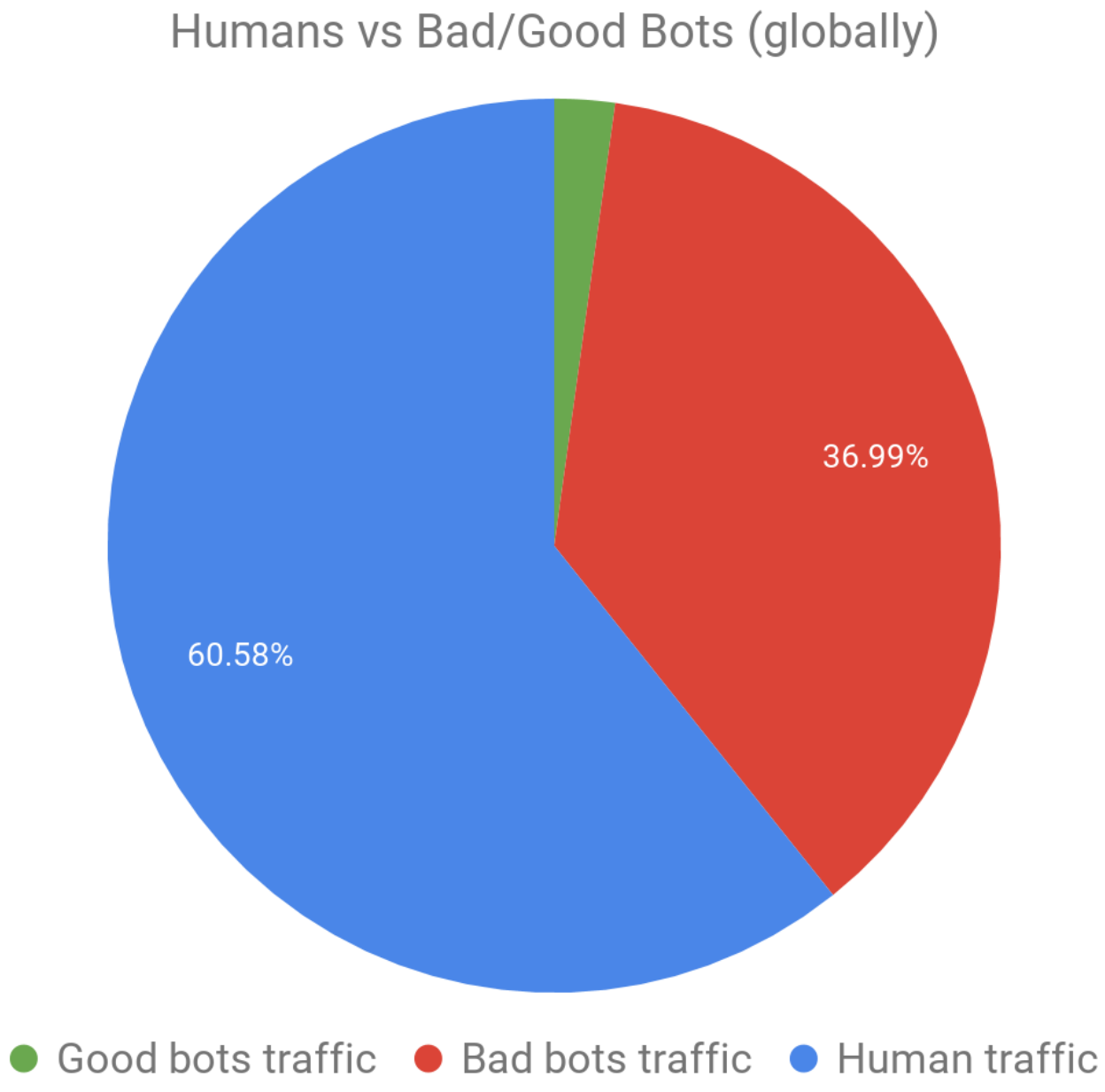
- **Feedback loop** - availability of the score on every request along with all features has tremendous value for continuous improvement of our detection mechanisms.

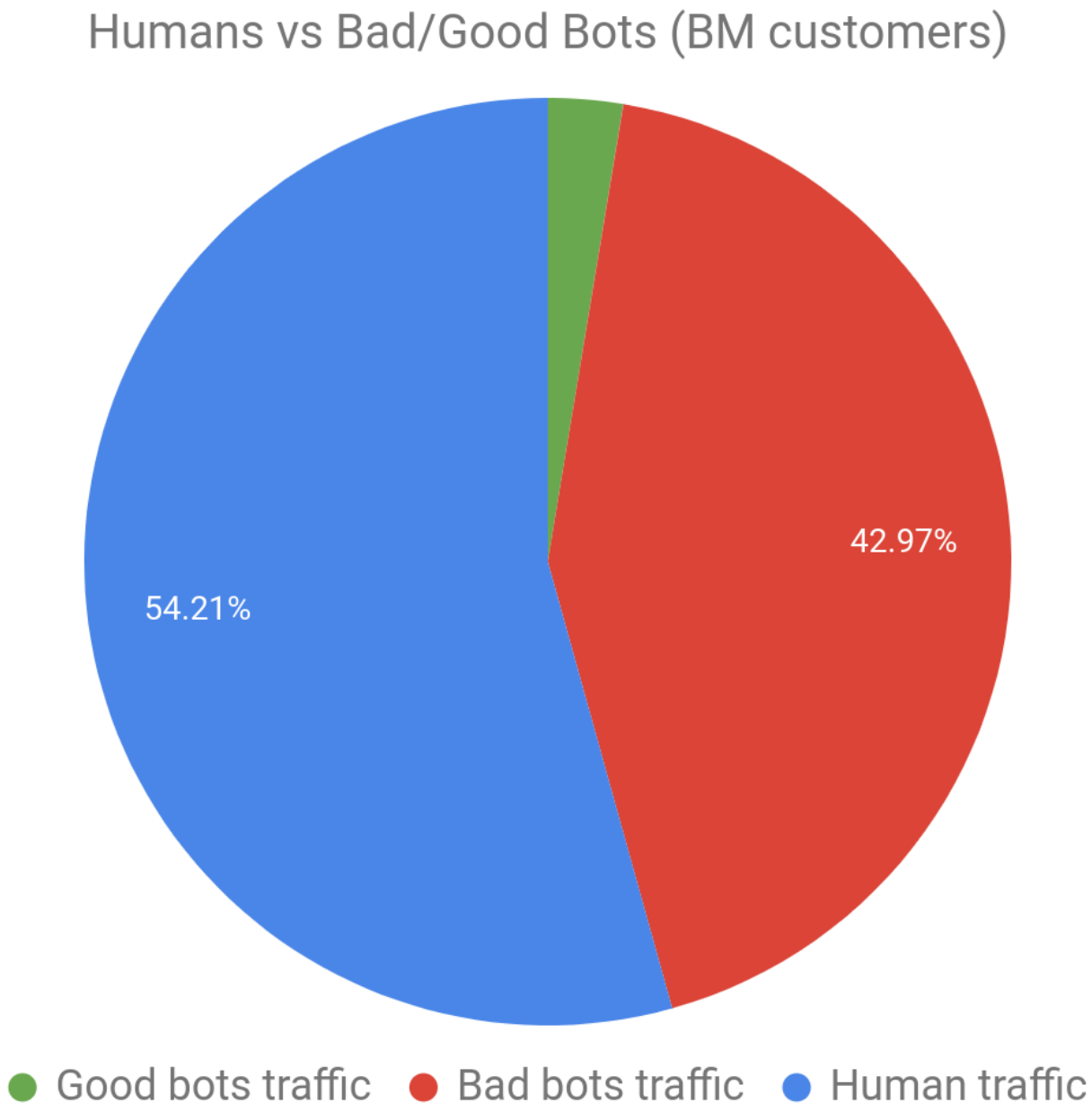
- **Ensures scaling** - if we can compute for score every request and customer, it means that every Internet property behind Cloudflare is a

potential Bot Management customer.

- **Global bot insights** - Cloudflare is sitting in front of more than 26M+ Internet properties, which allows us to understand and react to the tectonic shifts happening in security and threat intelligence over time.

Overall globally, more than third of the Internet traffic visible to Cloudflare is coming from bad bots, while Bot Management customers have the ratio of bad bots even higher at ~43%!





Let’s dive into specific detection mechanisms in chronological order of their integration with Cloudflare Bot Management.

Machine learning

The majority of decisions about the score are made using our machine learning models. These were also the first detection mechanisms to produce a score and to on-board customers back in 2018. The successful application of machine learning requires data high in Quantity, Diversity, and Quality, and

thanks to both free and paid customers, Cloudflare has all three, enabling continuous learning and improvement of our models for all of our customers.

At the core of the machine learning detection mechanism is CatBoost - a high-performance open source library for gradient boosting on decision trees. The choice of CatBoost was driven by the library's outstanding capabilities:

- **Categorical features support** - allowing us to train on even very high cardinality features.

- **Superior accuracy** - allowing us to reduce overfitting by using a novel gradient-boosting scheme.

- **Inference speed** - in our case it takes less than 50 microseconds to apply any of our models, making sure request processing stays extremely fast.

- **C and Rust API** - most of our business logic on the edge is written using Lua, more specifically LuaJIT, so having a compatible FFI interface to be able to apply models is fantastic.

There are multiple CatBoost models run on Cloudflare's Edge in the shadow mode on *every request on every machine*. One of the models is run in active mode, which influences the final score going to Firewall. All ML detection results and features are logged and recorded in ClickHouse for further analysis, model improvement, analytics and customer facing logs. We feed both categorical and numerical features into our models, extracted from request attributes and inter-request features built using those attributes, calculated and delivered by the *Gagarin* inter-requests features platform.

We're able to deploy new ML models in a matter of seconds using an extremely reliable and performant Quicksilver configuration database. The same mechanism can be used to configure which version of an ML model should be run in active mode for a specific customer.

A deep dive into our machine learning detection mechanism deserves a blog post of its own and it will cover how do we train and validate our models on trillions of requests using GPUs, how model feature delivery and extraction works, and how we explain and debug model predictions both internally and externally.

Heuristics engine

Not all problems in the world are the best solved with machine learning. We can tweak the ML models in various ways, but in certain cases they will likely underperform basic heuristics. Often the problems machine learning is trying to solve are not entirely new. When building the Bot Management solution it became apparent that sometimes a single attribute of the request could give a bot away. This means that we can create a bunch of simple rules capturing bots in a straightforward way, while also ensuring lowest false positives.

The heuristics engine was the second detection mechanism integrated into the Cloudflare Bot Management platform in 2019 and it's also applied on every request. We have multiple heuristic types and hundreds of specific rules based on certain attributes of the request, some of which are very hard to spoof. When any of the requests matches any of the heuristics - we assign the lowest possible score of 1.

The engine has the following properties:

- **Speed** - if ML model inference takes less than 50 microseconds per model, hundreds of heuristics can be applied just under 20 microseconds!

- **Deployability** - the heuristics engine allows us to add new heuristic in a matter of seconds using Quicksilver, and it will be applied on every request.

- **Vast coverage** - using a set of simple heuristics allows us to classify ~15% of global traffic and ~30% of Bot Management customers' traffic as bots. Not too bad for a few if conditions, right?

- **Lowest false positives** - because we're very sure and conservative on the heuristics we add, this detection mechanism has the lowest FP rate among all detection mechanisms.

- **Labels for ML** - because of the high certainty we use requests classified with heuristics to train our ML models, which then can generalize behavior learnt from from heuristics and improve detections accuracy.

So heuristics gave us a lift when tweaked with machine learning and they contained a lot of the intuition about the bots, which helped to advance the Cloudflare Bot Management platform and allowed us to onboard more customers.

Behavioral analysis

Machine learning and heuristics detections provide tremendous value, but both of them require human input on the labels, or basically a teacher to distinguish between right and wrong. While our supervised ML models can generalize well enough even on novel threats similar to what we taught them on, we decided to go further. What if there was an approach which doesn't require a teacher, but rather can learn to distinguish bad behavior from the normal behavior?

Enter the behavioral analysis detection mechanism, initially developed in 2018 and integrated with the Bot Management platform in 2019. This is an unsupervised machine learning approach, which has the following properties:

- **Fitting specific customer needs** - it's automatically enabled for all Bot Management customers, calculating and analyzing normal visitor

behavior over an extended period of time.

- **Detects bots never seen before** - as it doesn't use known bot labels, it can detect bots and anomalies from the normal behavior on specific customer's website.

- **Harder to evade** - anomalous behavior is often a direct result of the bot's specific goal.

Please stay tuned for a more detailed blog about behavioral analysis models and the platform powering this incredible detection mechanism, protecting many of our customers from unseen attacks.

Verified bots

So far we've discussed how to detect bad bots and humans. What about good bots, some of which are extremely useful for the customer website? Is there a need for a dedicated detection mechanism or is there something we could use from previously described detection mechanisms? While the majority of good bot requests (e.g. Googlebot, Bingbot, LinkedInbot) already have low score produced by other detection mechanisms, we also need a way to avoid accidental blocks of useful bots. That's how the Firewall field *cf.bot_management.verified_bot* came into existence in 2019, allowing customers to decide for themselves whether they want to let all of the good bots through or restrict access to certain parts of the website.

The actual platform calculating Verified Bot flag deserves a detailed blog on its own, but in the nutshell it has the following properties:

- **Validator based approach** - we support multiple validation mechanisms, each of them allowing us to reliably confirm good bot identity by clustering a set of IPs.

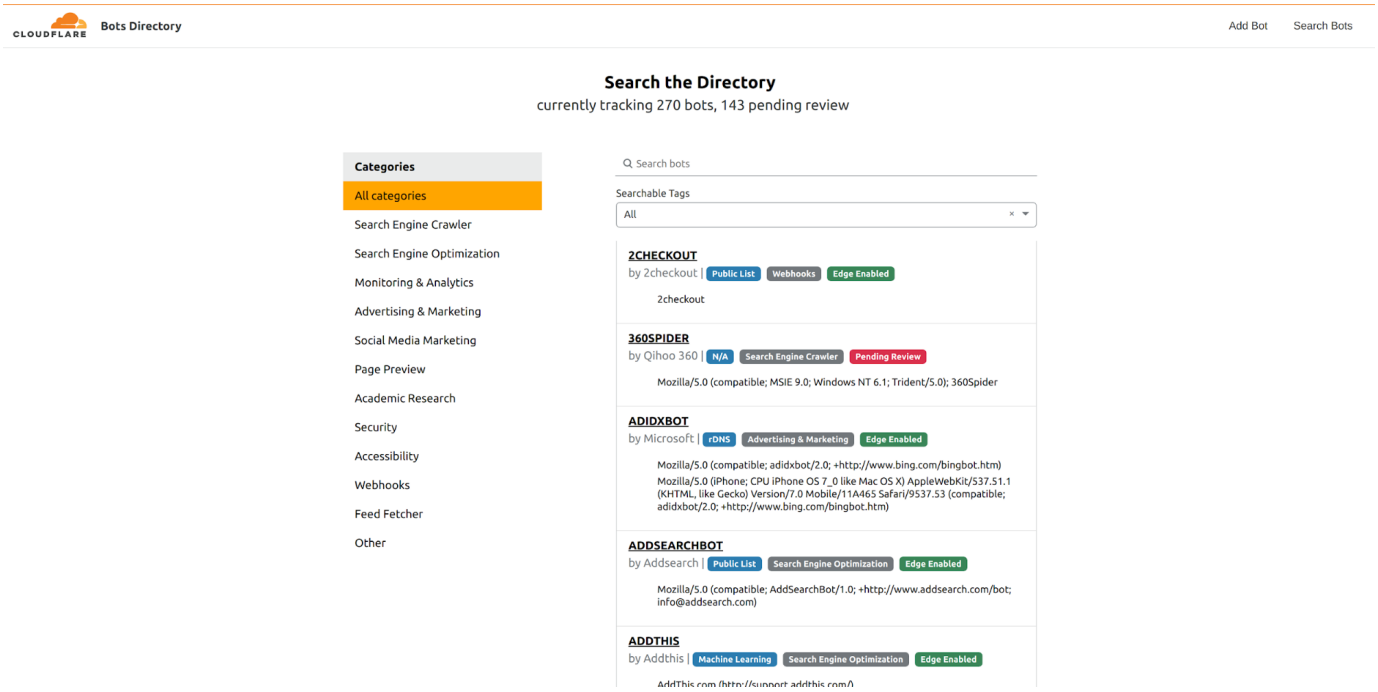
- **Reverse DNS validator** - performs a reverse DNS check to determine whether or not a bots IP address matches its alleged hostname.

- **ASN Block validator** - similar to rDNS check, but performed on ASN block.

- **Downloader validator** - collects good bot IPs from either text files or HTML pages hosted on bot owner sites.

- **Machine learning validator** - uses an unsupervised learning algorithm, clustering good bot IPs which are not possible to validate through other means.

- **Bots Directory** - a database with UI that stores and manages bots that pass through the Cloudflare network.



Bots directory UI sample

Using multiple validation methods listed above, the Verified Bots detection mechanism identifies hundreds of unique good bot identities, belonging to

different companies and categories.

JS fingerprinting

When it comes to Bot Management detection quality it's all about the signal quality and quantity. All previously described detections use request attributes sent over the network and analyzed on the server side using different techniques. Are there more signals available, which can be extracted from the client to improve our detections?

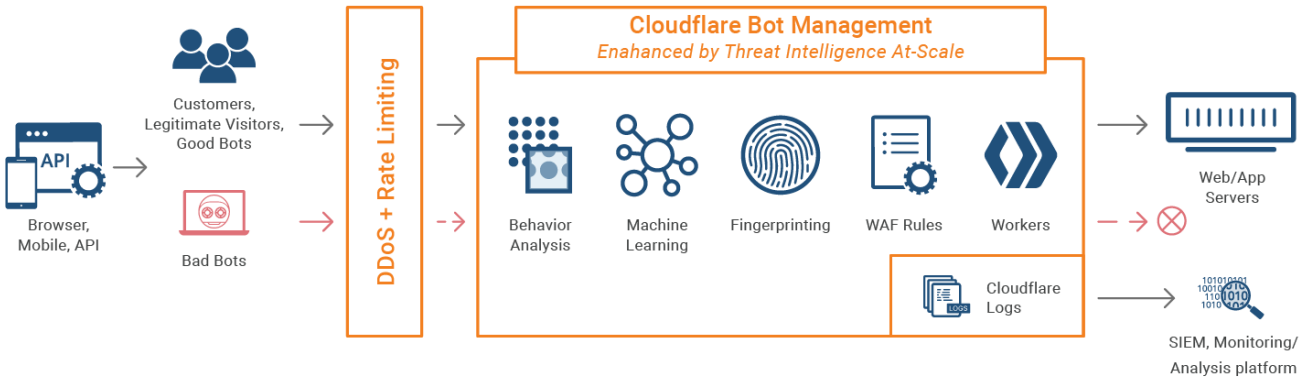
As a matter of fact there are plenty, as every browser has unique implementation quirks. Every web browser graphics output such as canvas depends on multiple layers such as hardware (GPU) and software (drivers, operating system rendering). This highly unique output allows precise differentiation between different browser/device types. Moreover, this is achievable without sacrificing website visitor privacy as it's not a supercookie, and it cannot be used to track and identify individual users, but only to confirm that request's user agent matches other telemetry gathered through browser canvas API.

This detection mechanism is implemented as a challenge-response system with challenge injected into the webpage on Cloudflare's edge. The challenge is then rendered in the background using provided graphic instructions and the result sent back to Cloudflare for validation and further action such as producing the score. There is a lot going on behind the scenes to make sure we get reliable results without sacrificing users' privacy while being tamper resistant to replay attacks. The system is currently in private beta and being evaluated for its effectiveness and we already see very promising results. Stay tuned for this new detection mechanism becoming widely available and the blog on how we've built it.

This concludes an overview of the five detection mechanisms we've built so far. It's time to sum it all up!

Summary

Cloudflare has the unique ability to collect data from trillions of requests flowing through its network every week. With this data, Cloudflare is able to identify likely bot activity with Machine Learning, Heuristics, Behavioral Analysis, and other detection mechanisms. Cloudflare Bot Management integrates seamlessly with other Cloudflare products, such as WAF and Workers.



All this could not be possible without hard work across multiple teams! First of all thanks to everybody on the Bots Team for their tremendous efforts to make this platform come to life. Other Cloudflare teams, most notably: Firewall, Data, Solutions Engineering, Performance, SRE, helped us a lot to design, build and support this incredible platform.

Bots team during Austin team summit 2019 hunting bots with axes :)

Lastly, there are more blogs from the Bots series coming soon, diving into internals of our detection mechanisms, so stay tuned for more exciting stories about Cloudflare Bot Management!

Deep Dive | Bot Management | Bots | Architecture | Machine Learning

RELATED POSTS

March 05, 2020 1:00PM

The History of the URL

On the 11th of January 1982 twenty-two computer scientists met to discuss an issue with ‘computer mail’ (now known as email). Attendees included the guy who would create Sun

Microsystems, the guy who made Zork, the NTP guy, and the guy who convinced the government to pay for Unix....

By Zack Bloom

Deep Dive, History

March 02, 2020 1:00PM

When Bloom filters don't bloom

Last month finally I had an opportunity to use Bloom filters. I became fascinated with the promise of this data structure, but I quickly realized it had some drawbacks. This blog post is the tale of my brief love affair with Bloom filters....

By Marek Majkowski

Deep Dive, Hardware, Optimization, Programming, Tools

December 05, 2019 7:28AM

Thinking about color

An exploration of building accessible color systems for flexible UI theming....

By Sam Mason de Caires, Adam Morse

Deep Dive, Design, Dashboard