Reblaze

2019 State of Bot Protection

Tzury Bar Yochay
Spiros Psarris
Tamara Shostak
Daniel Dekalo
Idan Yatziv
Yaniv Yagolnitzer

Reblaze Technologies LTD

This report is brought to you by Reblaze. It reflects the traffic conditions being experienced by our customers in 2019, the current state of the art in bot protection, and several areas of current research by our scientists into new capabilities and human/bot identification algorithms.

Table of Contents

Introduction

Web Traffic Composition

Robust bot management is essential for web security today. On average across different verticals, only 38 percent of incoming requests originate from human users. The remaining 62 percent have an automated source.

Not all bots are harmful. Some (such as search engine spiders) are often welcome, and some (such as content aggregators) are not overtly hostile. **However, almost 40 percent of incoming requests come from malicious bots.**

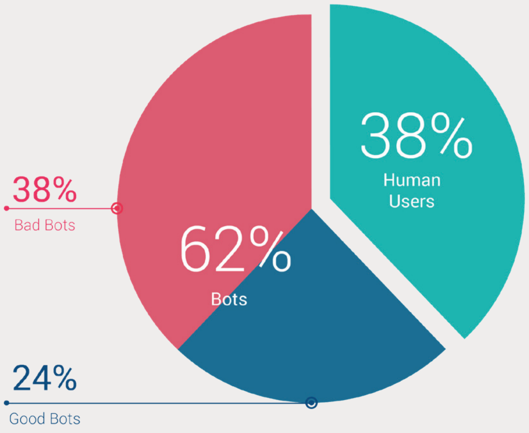
Consequences of Inadequate Protection

Threat actors use bots to wage a variety of web attacks. In fact, almost all attacks involve bots in one way or another. Hostile bots which are not identified and blocked can create a variety of problems for organizations with significant web assets (sites and web applications, microservices, and mobile/native API endpoints). Some of the potential problems are:

- **Site downtime**
  when DDoS attacks exhaust app/server resources.
- **Data theft**
  from scraping.
- **Site breaches**
  via vulnerability discovery bots.
- **Loss of revenue**
  from inventory hoarding.
- **Bad business decisions**
  from skewed analytics.
- **Account theft**
  from credential stuffing.
- **Merchant account problems**
  from credit card testing.
- **Degraded customer experience**
  from loss of bandwidth & resources.

Average traffic composition across verticals

Source: traffic processed by Reblaze
(more than four billion http/s requests per day).



Threat Landscape

Hostile bot traffic is not uniform across verticals. Different industries present different types of opportunities for threat actors. Thus, verticals tend to experience different types of attacks.

Often, when mitigating an attack, it's helpful to understand its source. A useful approach is the same logic that's become a cliche in Hollywood crime dramas, when a detective or investigator says "follow the money." Applying the same reasoning to cyberattacks is useful, and is expressed in the listings below by the acronym WSTG: "Who Stands To Gain."

The most common bot threats are shown below, and are discussed in more detail in the following pages.

> "
>
> WSTG: "Who Stands To Gain."
>
> Are the perpetrators most likely to be criminals or competitors?
>
> "

- DDoS
- Inventory Hoarding
- Vulnerability Scans
- Credit/Gift Card Fraud
- Scraping & Data Theft
- Advertising Abuse
- Credential Attacks
- Spam
- Application Abuse

Distributed Denial of Service (DDoS)

DDoS is the most dramatic, and probably the most feared, form of bot attack. Malicious actors use large networks of bots to create coordinated attacks at massive scale.

The goal is to disrupt the targeted organization by overwhelming its web applications or APIs with incoming requests, making them unavailable for normal use. If the victim cannot filter out the attack traffic, the disruption will last for as long as the attacker wishes.

DDoS has some important differences from other types of bot attacks. Many of the others can be waged by human threat actors. (For example, vulnerability testing can be done manually.) However, DDoS is a distributed attack that consists of automated traffic. Also, the bots behind a DDoS can vary widely in their sophistication: everything from complex malware running on zombie PCs, to simple IoT devices that have been hijacked for a mass attack.

Timing

DDoS attacks from competitors are often timed carefully. (For example, it's common in online gaming for a DDoS to occur right before a large race or sporting event. Bettors who are placing wagers will go to the first available platform.) Criminals often focus on longer periods, e.g. DDoS extortion of eCommerce sites during holiday shopping season.

WSTG

Criminals, competitors, occasionally governments.

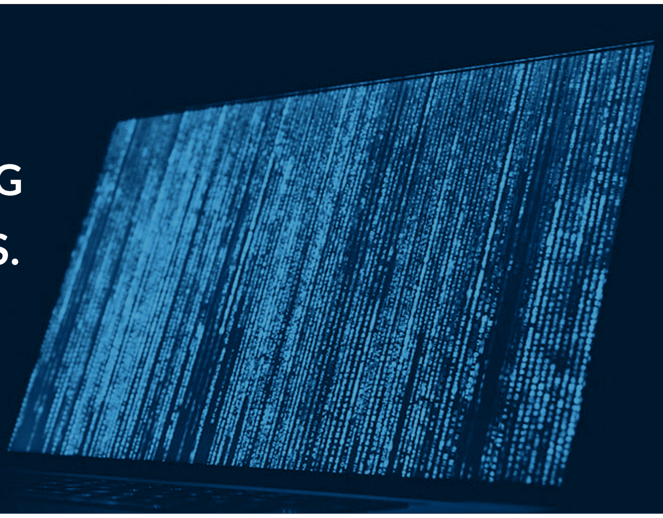
Motive

Competitors wish to disrupt the victim's business operations, making it impossible to serve customers. Criminals will often send ransom demands (i.e., DDoS extortion). Hackers will use DDoS to get revenge for perceived injustices. Governments will use DDoS for suppressing opposing political viewpoints (of humanitarian groups, journalism/media sources, etc.).

Consequences

Loss of sales revenue, and longer-term loss of customer goodwill and reputation in the marketplace. In extreme or frequent cases, a decline in search engine rankings can occur.

A GROWING NUMBER OF DDOS ATTACKS ARE BEING AIMED AT API ENDPOINTS.



Credit Card Fraud

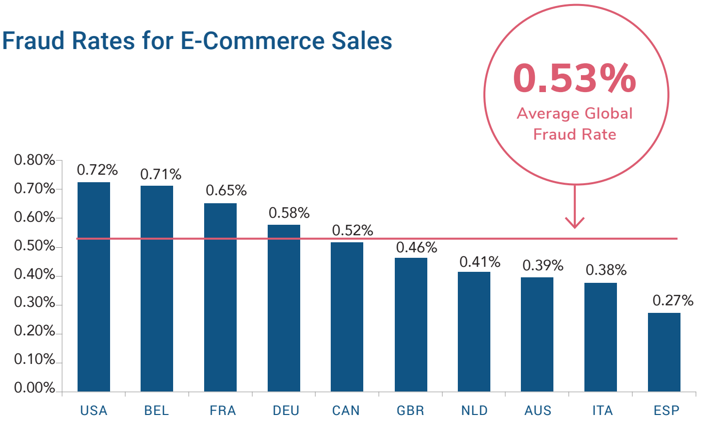
Bots are the foundation of a card criminal's arsenal. They are used in a variety of methods to obtain or validate stolen card numbers. Later, the numbers are used fraudulently, which results in lost revenue and chargebacks to the unfortunate merchant.

To steal card data, bots scan for vulnerabilities within retailers and other sites that process payments. When a vulnerability is found, the hacker breaches the site and steals the data. One successful attack can produce a windfall of cards: thousands, or even tens of thousands of active numbers.

Threat actors also use bots to validate stolen card numbers. Bots enter the numbers into web applications to see if they are accepted or rejected. A similar technique is used to discover new cards: bots cycle through potential numbers and enter them into web applications. This is a crude, but effective, way to steal additional cards that were previously unknown to the attacker.

The scale of online credit card abuse is illustrated by the prevalence of "card not present" fraud. This is growing, thanks in part to the rise of EMV chip cards. EMV makes physical card fraud more difficult, which discourages criminals from monetizing stolen numbers by printing physical cards. Thus, more criminals are moving online to monetize their stolen numbers.

Fraud Rates for E-Commerce Sales



Source: "Card-Not-Present Fraud Around the World," US Payments Forum

WSTG

Criminals

Motive

Selling hijacked numbers on the darknet and/or using them to make fraudulent purchases.

Consequences

Lost revenue when products are shipped and fraudulent payments are subsequently reversed. Card fraud also results in penalties and (eventually) account cancellations from merchant account providers.

77%

OF MERCHANTS IN THE UNITED STATES ARE ONLINE. THIS IS THE LARGEST PERCENTAGE IN THE WORLD.

Gift Card Fraud

(including Loyalty/Reward accounts)

Criminals steal gift cards by using bots to stuff possible numbers into applications until valid ones are found. Validated card numbers are used to purchase goods, or are sold for cash through various online services.

Criminals can use similar methods to perform coupon code discovery. While not as outright fraudulent as the above, it still has a direct impact on revenue.

Bots also use credential stuffing (see Credential Attacks, below) to take over loyalty/reward accounts, and drain their balances—potentially extracting funds from customers' linked debit cards.

Threat actors have proven quite creative in exploiting gift and loyalty programs. Past examples included the discovery of programming errors in certain gift-card account APIs, creating potential race conditions. To exploit them, bots would submit simultaneous transfers among multiple cards, sometimes resulting in funds being credited to one card without being debited from another. Fraudsters were able to convert a small "seed" of initial funds into large gift-card balances.

WSTG

Criminals

Motive

Stealing or fraudulently creating gift-card balances for resale and/or purchases.

Consequences

Reduced revenue and damaged profit margins. Loyalty and reward program abuse results in harmful publicity and customer dissatisfaction (especially when customers have personal debit cards attached to their accounts).

RECENT VICTIMS OF LOYALTY AND REWARD PROGRAM ATTACKS INCLUDE STARBUCKS, SUBWAY, HILTON, AND STARWOOD.

Credential Attacks

(Enumeration/Brute-Force, Credential Stuffing, Account Creation or Takeover)

User credentials are highly coveted commodities in the dark web. Hackers discover credentials by sending out bots to wage brute-force attacks; the bots attempt to gain access to a web application by trying every possible combination of letters, numbers, and symbols, to see which combinations work. Or, they steal credential sets (personal identification data, account logins and passwords, contact data, etc.) in massive data breaches.

Valid credentials can then be used in a variety of cyberattacks, and can also be sold in illicit marketplaces for others to use. Credentials can allow attackers to take over the affected accounts within the targeted web application. Another common attack is to use bots to "stuff" the credentials into the login pages of many other web applications (especially high-value targets like bank websites, payment providers, and so on). Unfortunately, many people still use the same credentials across a variety of websites. Therefore, credential stuffing allows an attacker to leverage a single data breach into the successful takeover of multiple accounts across different websites.

WSTG

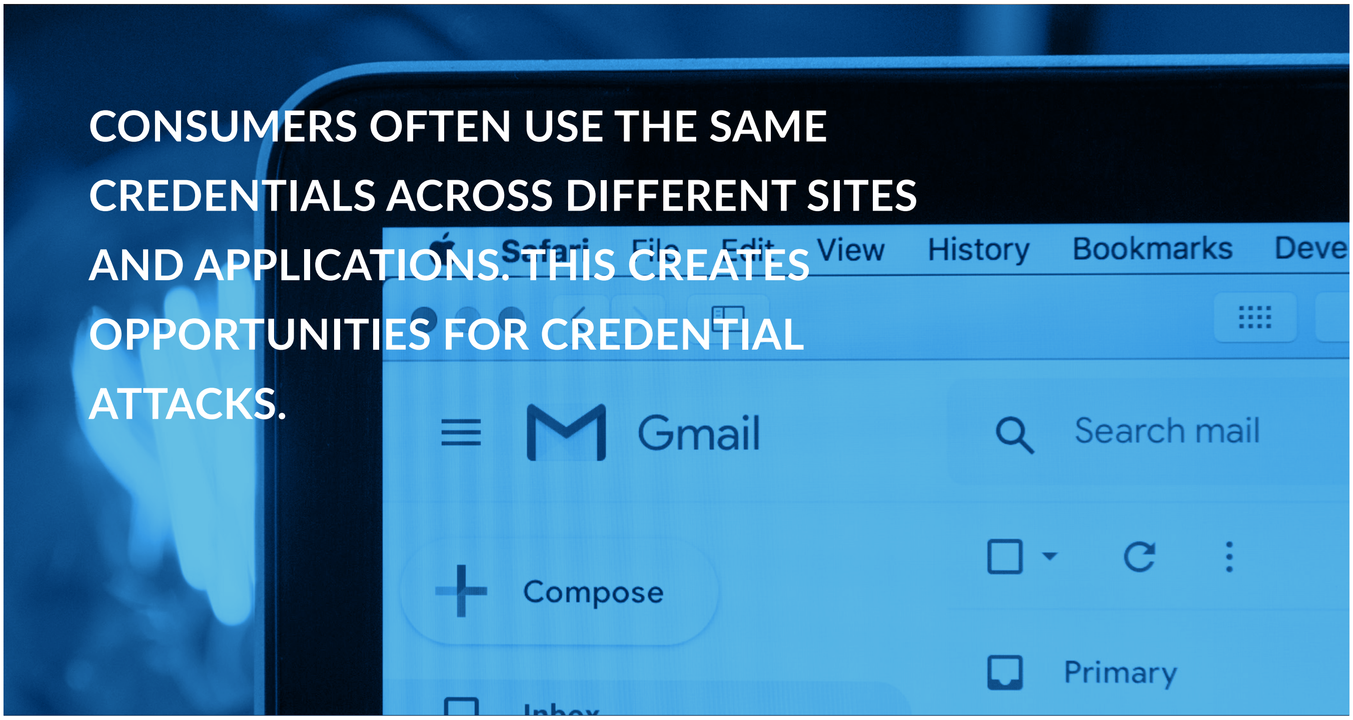
Criminals

Motive

Obtaining credentials for resale, abuse, or both.

Consequences

Hijacked accounts cause numerous problems for the victim and its customers. When the data breaches are discovered, the victim is the target of bad publicity, loss of reputation and trust, and might receive fines and penalties from industry and privacy regulators.



CONSUMERS OFTEN USE THE SAME CREDENTIALS ACROSS DIFFERENT SITES AND APPLICATIONS. THIS CREATES OPPORTUNITIES FOR CREDENTIAL ATTACKS.

Inventory Hoarding

Web applications which offer online purchasing or reservations are vulnerable to inventory hoarding (a.k.a. "Denial of Inventory"), when hostile bots make inventory unavailable to legitimate customers. Example: bots attack retail sites by adding products to shopping carts, but never completing the purchases.

In some industries, inventory hoarding attacks occur frequently. For example, travel sites and applications are often attacked by bots that abuse time-to-checkout policies (which usually allow 15 minutes or so for customers to complete their transactions), continually looping and booking reservations without ever purchasing tickets.

This obviously prevents actual customers from purchasing, but the financial damage can be far worse. Travel sites and applications often get their data from aggregators. Each time a “customer” searches for flights, a small financial liability (a data lookup fee) is created. If the customer buys a ticket, the aggregator gets a commission on the sale; otherwise, the fee is charged. Since bots never buy tickets, their continual data requests can accrue significant expenses for site owners.

WSTG

Competitors

Motive

Commercial harm to the victim.

Consequences

In effect, inventory hoarding is an Application-Layer Denial-of-Service attack. It can result in direct loss of revenue, because legitimate customers cannot make purchases. Products which expire (e.g., tickets to an event) can go unsold. Sellers can accrue expenses such as data-lookup fees. Consumer goodwill and trust can be damaged.

BEFORE DEPLOYING REBLAZE, A CERTAIN INTERNATIONAL AIRLINE WAS LOSING $600,000 ANNUALLY IN GDS FEES FROM HOARDING BOTS.

Scraping and Data Theft

(Prices, Content, etc.)

Scraper bots steal data from online sources. This is commonly seen in verticals such as data aggregators and other providers which gather or generate data and content, and then sell access to it. Scraping is obviously a direct threat to these business models.

Elsewhere, scraping can cause indirect damage. For example, retail sites contain prices and other product data which, when stolen, can destroy a competitive advantage.

Sophisticated scraper bots can eventually steal entire databases, even when they aren't directly available to users of the targeted site or application. This can be done through repeated queries or requests, using different parameters each time. For example, insurance companies provide rate quotes for specific combinations of input criteria. A scraper bot can submit continual requests for quotes, with a different combination of criteria each time, and capture the quotes that are returned. Eventually, the complete database of rates can be obtained.

WSTG

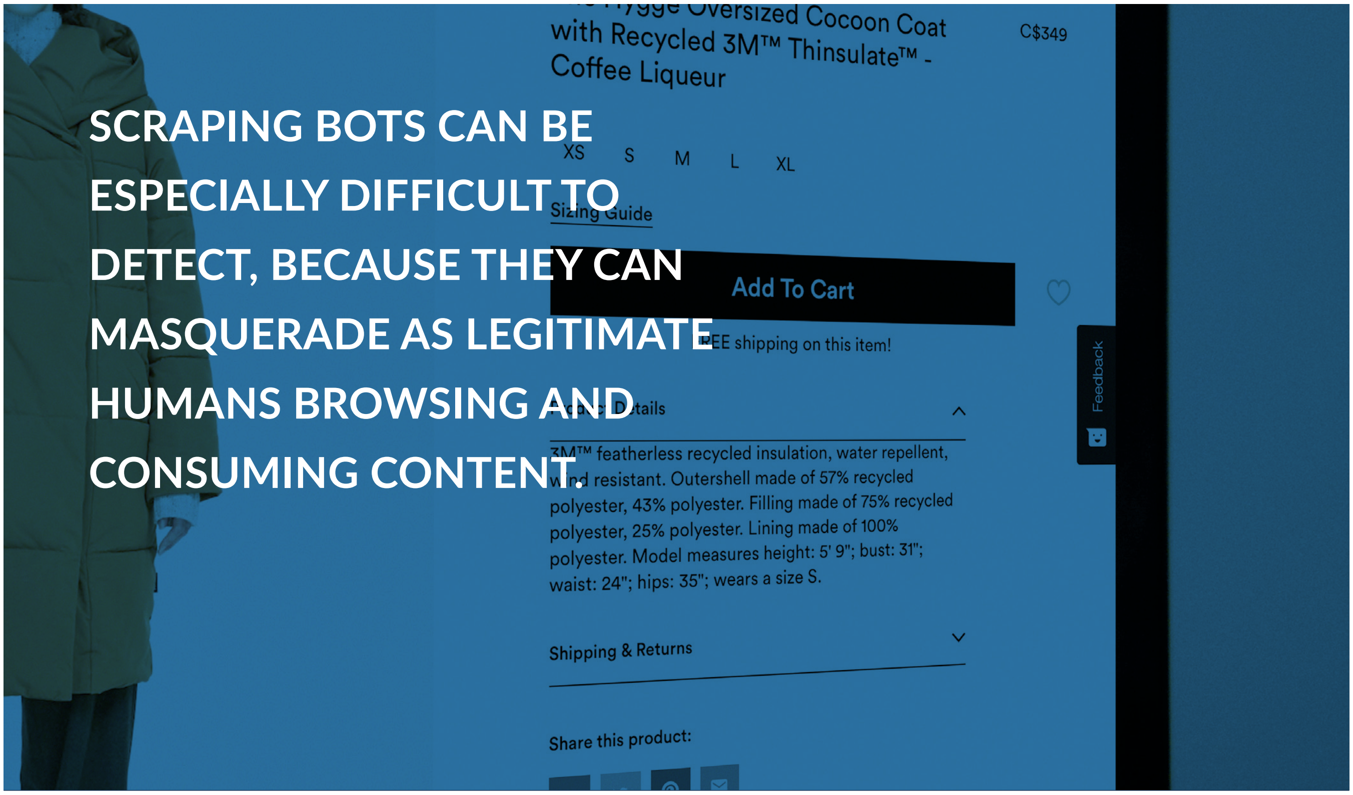
Competitors and criminals

Motive

Competitors wish to undercut the victim's prices and their sales. They can also steal useful content such as product reviews, boosting their own sales at the expense of the victim. Criminals steal commercially valuable data for resale.

Consequences

Degraded search engine rankings, damaged reputation among users, and declining user base.



SCRAPING BOTS CAN BE ESPECIALLY DIFFICULT TO DETECT, BECAUSE THEY CAN MASQUERADE AS LEGITIMATE HUMANS BROWSING AND CONSUMING CONTENT.

ygge Oversized Cocoon Coat with Recycled 3M™ Thinsulate™ - Coffee Liqueur

C$349

FREE shipping on this item!

3M™ featherless recycled insulation, water repellent, wind resistant. Outershell made of 57% recycled polyester, 43% polyester. Filling made of 75% recycled polyester, 25% polyester. Lining made of 100% polyester. Model measures height: 5' 9"; bust: 31"; waist: 24"; hips: 35"; wears a size S.

Spam

Sites that accept user-submitted content (posts, reviews, etc.) are usually assailed by bots leaving spam comments and links.

Some commercial site platforms offer protection against spam with built-in or add-on modules. However, this protection often relies on obsolete techniques such as CAPTCHA. (For a discussion of this, see the Traditional Detection Methods section later in this document.)

Even if this filtering works, it only means that the spam does not appear to legitimate site visitors. But the spam bots still harm the site, since the incoming spam content consumes bandwidth, requires compute resources to process, and also requires a potentially costly subscription to the spam-protection service.

Accepting spam content and then attempting to filter it is a partial solution at best. The best approach is to block incoming traffic from spam bots, so that their requests never even reach the targeted site. More on this later.

WSTG

Competitors and criminals

Motive

Illegitimately building search engine rankings and generating direct traffic for the spammer's sites.

Consequences

Search engines penalize sites that are polluted with spam, so the sites will appear lower in the search engine rankings. This reduces incoming traffic. As customers and users notice the spam, the site's reputation will also be damaged. Eventually, the user base will decline.

MANY ANTI-SPAMBOT TECHNIQUES (SUCH AS REQUIRING INTERACTIVE VERIFICATION OF NEW ACCOUNTS) CAN ALSO DECREASE THE NUMBER OF LEGITIMATE USERS.

Vulnerability Scans

Threat actors use bots to automatically scan large numbers of systems for known vulnerabilities. When an exploitable system is found, hackers follow up with direct attacks.

This follow-up activity can take a variety of forms, depending on the weakness that was discovered. Immediate attacks include data breaches, malware drops, and ransomware encryption attacks. For large networks that are perceived to have high long-term value, the attacker might install a backdoor instead, then use it to penetrate the network more deeply.

There are many examples of discoverable vulnerabilities being exploited. Perhaps the most severe and notorious recent security incident is the Equifax breach, which was made possible by an unpatched vulnerability in Apache Struts.

WSTG

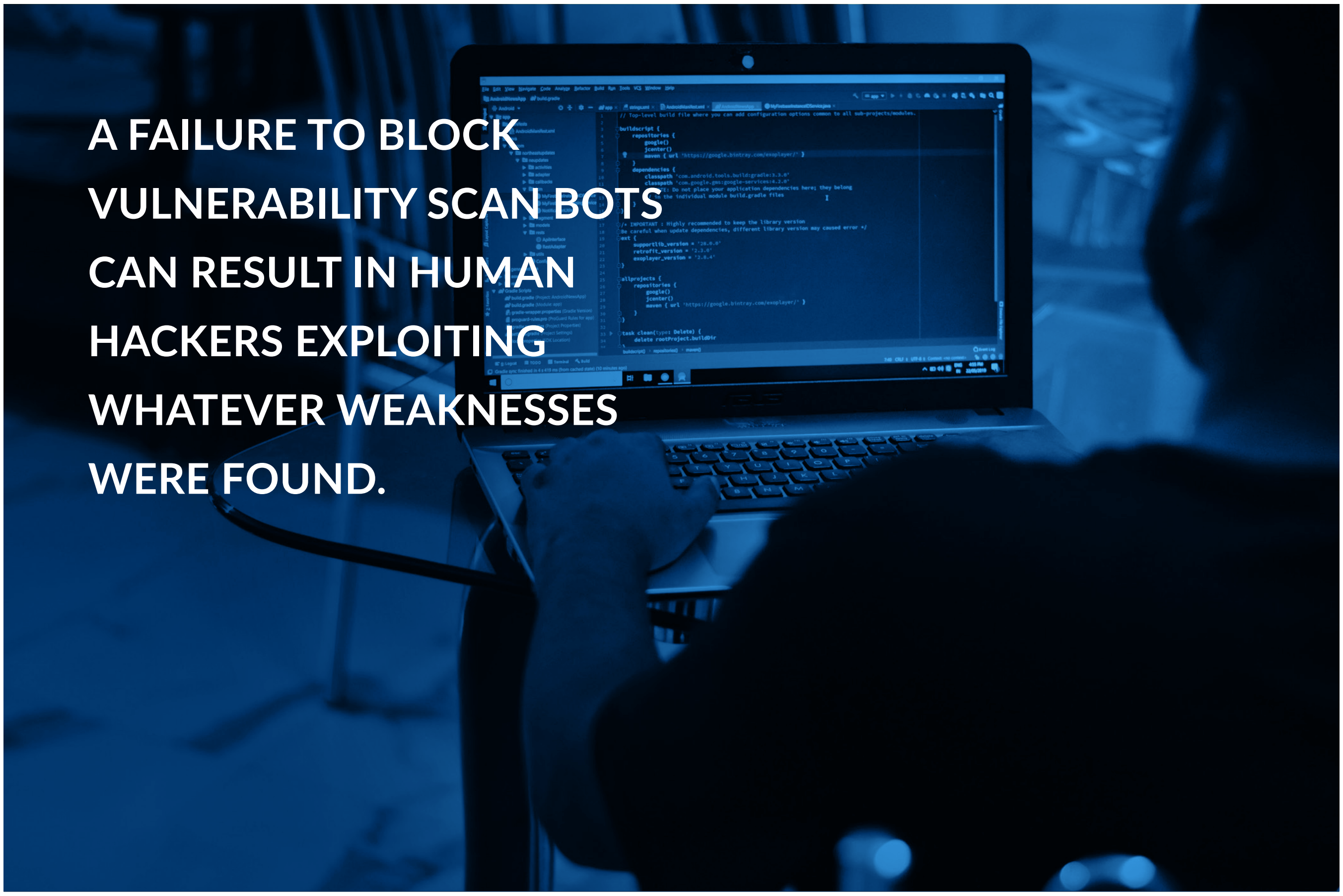
Criminals

Motive

Enabling many other forms of attacks.

Consequences

Large-scale data breaches, installation of backdoors, ransomware encryption attacks, and many other harmful events.



A FAILURE TO BLOCK VULNERABILITY SCAN BOTS CAN RESULT IN HUMAN HACKERS EXPLOITING WHATEVER WEAKNESSES WERE FOUND.

Advertising Abuse (Click fraud)

Advertising bot attacks sound benign, but they can cause significant damage. Click fraud occurs when bots are sent to “click” on ads; it can skew the results of a commercial or political ad campaign.

The direct victim is the advertiser who invests poorly and spends the ad budget in the wrong places. The site hosting the ads (and not properly controlling bot traffic) is harmed later, when the network blacklists it.

Advertising networks can be quite aggressive about blacklisting sites which have inadequate bot control. Otherwise, their advertisers slow down or stop their campaigns due to a failure to receive worthwhile results (despite getting lots of “clicks”).

Fraudulent clicks from bots often result in reversals of payments that were made earlier by the ad networks. For the site that hosted the ads, this results in loss of revenue. This loss is irrecoverable, because those page views are in the past, and the opportunity to receive ad revenue from them cannot be repeated.

WSTG

Criminals and competitors

Motive

Criminals generate revenue from affiliate commissions and fraudulent clicks on ads on their own web properties. Competitors wish to financially harm the victim.

Consequences

Lost income (after ad networks reverse payments from fraudulent clicks). Lost opportunity to generate revenue from future page views, because the networks will eventually refuse to supply ad inventory to the victim’s sites.

CLICK FRAUD IS A GROWING PROBLEM ON SOCIAL MEDIA ADVERTISING PLATFORMS.

Application Abuse

This includes a large variety of hostile bot activities that don't fall into the previous categories, where bots abuse specific capabilities of the victim's web application or API. For example, bots will exploit a phone system API to send out massive amounts of SMS spam.

API abuse is an increasingly large segment of this category. Many of the attacks discussed previously can be waged through APIs. Additionally, many applications offer their own specific opportunities for abuse.

Protecting API endpoints is a twofold problem. Usually, before threat actors deploy their specific attacks on an application, they first have to reverse-engineer its API.

Thus, a web security solution must prevent this from happening, along with the more obvious challenges of detecting API abuse, enforcing API schemas, and so on. More on this below.

WSTG

Usually criminals, occasionally competitors.

Motive

Varies.

Consequences

Varies. DDoS, credential attacks, inventory hoarding/denial, scraping & data theft, spam, and fraud are all common attacks that can be committed via an application or API.

AS MOBILE/NATIVE APPLICATIONS HAVE GROWN INCREASINGLY IMPORTANT, THREAT ACTORS ARE DEVOTING MORE TIME AND ATTENTION TO ABUSING THEIR BACKENDS.

Bots Threats Per Vertical

Hostile bot traffic is not uniform across verticals.

Different industries present different types of opportunities for threat actors. Thus, verticals tend to experience different types of attacks.

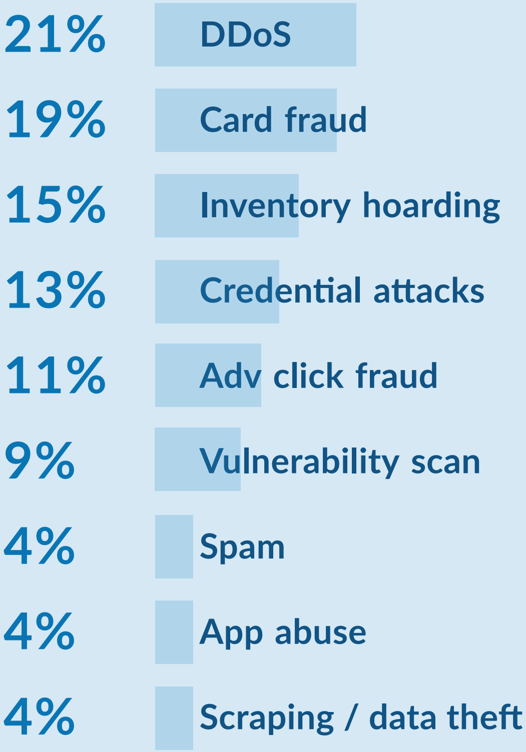
The statistics below reflect the composition of incoming traffic for Reblaze's client organizations and companies in some prominent industries.

01.Retail

Of all verticals, online retail tends to face the broadest range of bot-based threats. Ecommerce sites and applications present a rich array of illicit opportunities for threat actors.

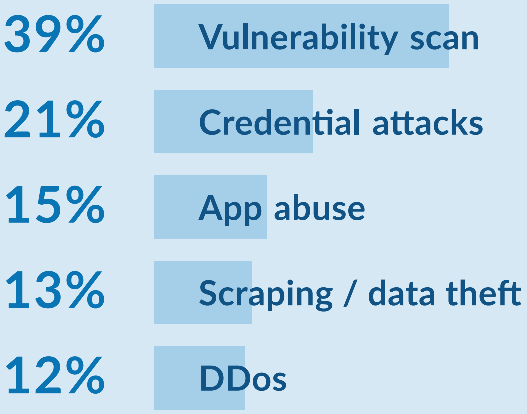
Retail sites and applications rely directly on incoming traffic for revenue; thus, DDoS extortion is potentially lucrative. Payments are processed: thus, credit card fraud is constantly being attempted. Customers must be able to access their accounts; thus, hackers have the ability to stuff credentials, and the incentive to breach the retailer's backend and steal account data. Customers can often upload their own content, such as reviews: thus, spam bots are rampant.

Sites and applications often have loyalty and/or gift card programs: thus, gift card fraud can be profitable. Inventory hoarding damages revenue, but is difficult to detect; thus, it is attractive to unscrupulous competitors who are eager to gain market share. And so on.



02.SaaS

SaaS providers face several major types of bot threats. Many SaaS platforms process and store large amounts of customer data. This makes them attractive targets for data theft, along with account & credential attacks. Most providers seek to remain competitive by continually adding features to their platforms; this creates an expanding attack surface, susceptible to application abuse (especially within APIs, which many security solutions cannot defend adequately). Also, attacks which makes a platform unresponsive to customers creates customer dissatisfaction, increases the churn rate, and damages the provider's reputation in the marketplace. From a cybercriminal's perspective, this creates a high perceived value for DDoS extortion.

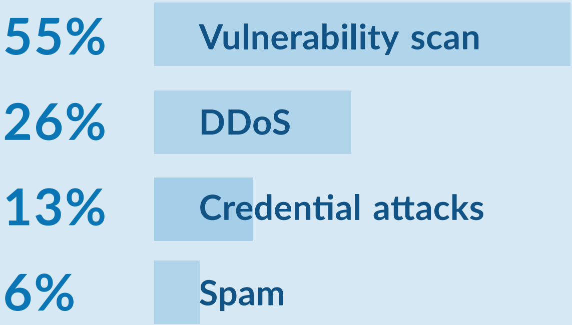


03.Education

Malicious bots are often used for several types of attacks against educational institutions. DDoS attacks are common: along with those that occur without an obvious motive, there are the usual attempts at extortion, revenge, and political grandstanding.

Vulnerability scans are also frequent. Educational institutions tend to be large public organizations. Therefore, many threat actors perceive (whether correctly or incorrectly) that their web assets are less likely to have effective protection against cyberattacks. A successful breach creates an opportunity for ransomware payoffs, along with data theft.

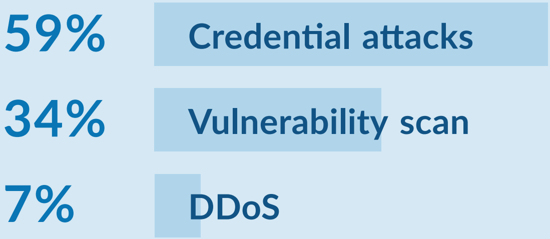
Educational institutions often store detailed PII (Personally Identifiable Information) for their students. Therefore, attackers are especially interested in stealing student account data, whether by direct data theft or through credential attacks.



04.Healthcare

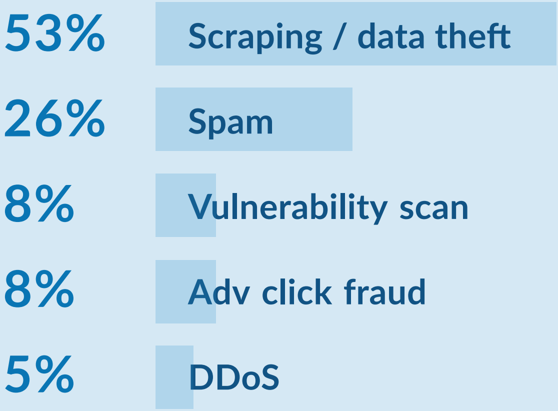
Bot-based attacks create unique challenges for healthcare organizations. No other industry has such strict legal and regulatory requirements for maintaining tight data security, or such potentially damaging consequences when security measures fail. Account and credential attacks are a major threat; a compromise of patient data can result in punitive fines and penalties.

Vulnerability scans can result in system breaches; subsequent ransomware attacks can hinder the healthcare provider from providing effective care for the patients.



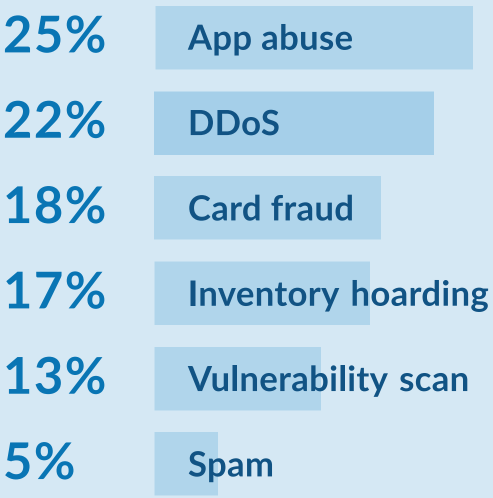
05.Content/Data Aggregators

In this vertical, sites and applications sell access to content and/or intellectual property. This can include a wide variety of content: everything from databases of legal records and court cases, to aggregations of current real estate price valuations. Unlike some other verticals, these platforms often do not attract many payment-related attacks such as credit card fraud. (Many content platforms do not process funds directly: they rely on third-party services for subscription payments and so on.) Instead, cybercriminals are most interested in stealing the data itself. Scraper bots masquerade as legitimate users, copying data that is publicly accessible. Other bots attempt to penetrate or otherwise circumvent security measures, to steal data that is meant to be restricted. Spam bots pollute user-generated content (e.g., marketplace reviews) with SEO link drops, or corrupt it with self-promotional content.



06.Travel and Ticketing

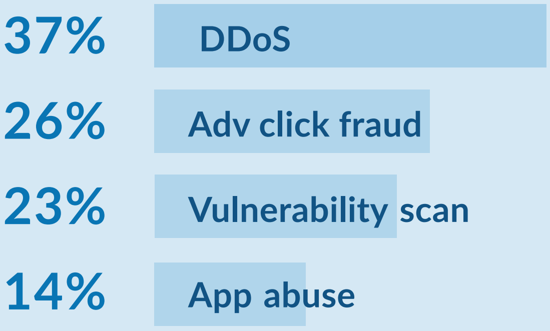
Companies within the online ticketing/booking vertical must defend against a variety of bot-based threats. Because these platforms sell directly B2C online, attempted credit card fraud is a constant problem. DDoS attacks are also frequent. These sites and applications also must contend with several threats that are more specific to their industry. For example, inadequate bot protection will leave ticketing/booking platforms vulnerable to inventory hoarding. As mentioned previously, hoarding bots can not only reduce reservations and ticket sales (because they prevent legitimate customers from buying), they can also increase expenses (due to data fees that many platforms pay). Application-specific abuse is also a growing problem, thanks to the increasing popularity and feature-richness of mobile/native applications for travel and ticketing/booking.



07.Gaming

Gaming sites and applications accept and pay out large amounts of money, while operating in loosely-regulated jurisdictions. This makes them extremely popular targets for cybercriminals. DDoS attacks on gaming platforms are rampant: not only continuous attacks for extortion, but also intermittent attacks from competitors to knock down a platform right before important events (playoff games, prominent races, etc.) when a large rush of last-minute bets would otherwise be coming in.

Other bot attacks include vulnerability scans (to facilitate system breaches), advertising click fraud (because many of the ad networks which accept gaming ads do not have robust anti-fraud controls), and application abuse.

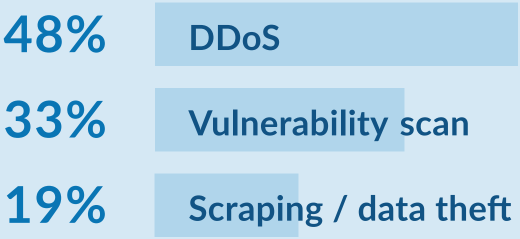


08.Government

Attacks on government web assets can vary, depending on the size of the agency. The types of attacks are generally the same: governments are most often threatened by DDoS, data theft, and vulnerability scans. But threat actors often wage them differently for different targets.

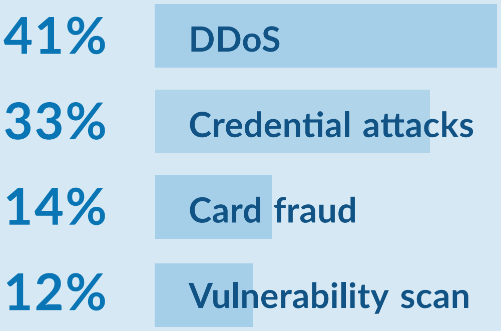
This is especially true for DDoS. Attacks on the largest jurisdictions, such as national agencies, are generally politically motivated. An example of this is “OpIsrael,” the annual coordinated attacks on Israeli sites by anti-Israel activists around the world. In contrast, attacks on smaller targets (such as local or municipal governments) are often waged for extortion instead.

Threat actors also use bots to probe government sites for vulnerabilities. Hackers frequently target government sites for ransomware attacks, due to a perception that these organizations are more likely to have inadequate or out-of-date defenses. Once a vulnerability is discovered, the attacker breaches the site and either steals data, encrypts it as part of an extortion attempt, or both.



09.FinTech

Bots are used in a variety of attacks against financial service providers. This includes everything from vulnerability scans to sophisticated account takeovers (e.g., credential stuffing combined with identity/IP rotation)—subtle attacks which are very difficult for most security solutions to detect, but with disastrous consequences for the provider if they succeed. Even a small security incident is enough to cause PR nightmares, punitive regulatory fines, and potentially crippling loss of business.



Traditional Detection Methods

Historically, bots have been detected with several different approaches.

Rate limiting

Monitoring consumption rates and blocking requestors above a certain threshold.

Blacklisting

Refusing incoming requests from IP addresses that are found on blacklists.

Signature recognition

Identifying patterns in the incoming requests that indicate that the originator is a bot.

Environment detection

JavaScript (JS) challenges and other tests such as cookie handling, to detect the absence of a normal web browser environment.

CAPTCHA/reCAPTCHA

[Completely Automated Public Turing test to tell Computers and Humans Apart]: Challenge-response puzzles presented to web visitors.

These approaches still work against older bots. However, problems arise when these are the *only* methods that an intrusion detection system (IDS) or bot management solution uses. Unfortunately, this is still often the case.

Why They Don't Work Against Modern Bots

There are several reasons why traditional methods have become inadequate. Some have arisen from the increasing sophistication of attackers, while others are the result of larger trends.

More sophisticated attackers

Threat actors are continually striving to make their tools more effective. Newer generations of bots are designed to avoid detection.

Rate limiting

is evaded by rotating IPs and keeping the rate of requests to 'reasonable' levels.

Blacklisting

is also avoided by IP rotation. The increasing irrelevance of IP tracking is discussed further below.

Signature recognition

is defeated by spoofing user agent strings and other deception, so that the bot appears to be a human user.

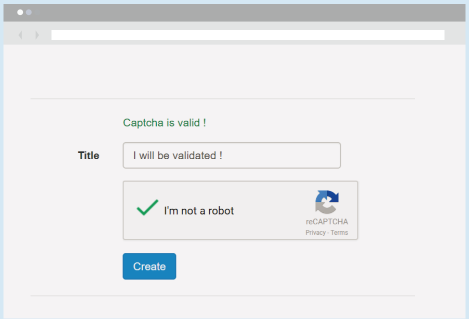
JS

Environment detection

can be evaded by many headless browsers (browsers run programmatically without a GUI). They can pretend to be "real" web browsers: they handle cookies, execute JavaScript, and so on.

Headless browsers in particular have become quite sophisticated. Because they have a number of legitimate uses, developers have created headless functionality in a variety of web languages and frameworks. Unfortunately, threat actors have taken advantage of this.

CAPTCHA /reCAPTCHA problems



Captcha is valid !

Title: I will be validated !

A well-known method for verifying human web users is CAPTCHA. These challenges are presented to dubious requestors; in theory, humans should be able to solve them, while bots cannot.

In reality, CAPTCHA has turned into an arms race. Researchers and threat actors are continually finding automated techniques for solving the challenges. As a result, several successive generations of CAPTCHA have been published, to increase the difficulty in solving them automatically.

Unfortunately, this also means that CAPTCHA and reCAPTCHA puzzles have created an increasingly negative user experience for sites that include them. Efforts like Google's "No CAPTCHA reCAPTCHAs" (shown above) have only partially mitigated this, since they frequently revert to full challenges.

A major development in this field was Google's 2018 release of reCAPTCHA v3. This promised bot detection "without user interaction." In other words, human users would no longer have to pass any on-screen challenges; all the bot detection would be done programmatically by Google behind the scenes. Unfortunately, reCAPTCHA v3 has two major flaws:

- **It doesn't accomplish its stated purpose.***
- **It has potential privacy issues, since it leverages Google's cookies in the user's browser. Since Google encourages site admins to place the reCAPTCHA code on all pages within a site, widespread adoption of v3 will allow Google to more thoroughly track the web usage of a large portion of Internet users. This has created some backlash and resistance to using it.**

The goal of CAPTCHA is very attractive: to install a single code snippet and achieve robust bot detection and exclusion. But that goal has not yet been achieved.

* *Google reCAPTCHA v3 can be solved by an automated process 97.4 percent of the time.*

The increasing irrelevance of IP and Geolocation

Traditional methods of tracking attackers are based on IP address. Today, an IDS which relies heavily on IP tracking has limited effectiveness.

The reasons for this are similar to those underlying Google's BeyondCorp initiative and the underlying Zero Trust Network model. It is the opposite of the old castle-and-moat approach; rather than treating users inside the perimeter as "good" and those outside as "bad," it scrutinizes all users, and grants/denies access to resources based on who they are.

Similarly, IP address is no longer a useful way of distinguishing hostile web attackers from legitimate users.

There is no longer a useful distinction between "good" IPs and "bad" IPs, because all users (whether hostile or legitimate) can have varying addresses.

For example, a legitimate user might access a web application over a mobile device, and its address and connection type (4G, LTE, etc.) can change multiple times during a session. Or perhaps the person is at an airport (or a coffee shop, or an airplane, or a library…) while trying to use a native app. The device is accessing the application API through public WiFi, and sharing a public IP with hundreds or even thousands of other Internet users that day. If an earlier user was hostile, and the API endpoint's WAF blacklisted the IP, then subsequent legitimate users will not have access.

Meanwhile, attackers are deliberately obscuring their IP-based identities. Threat actors are abusing cellular gateways, infecting IoT devices, distributing compromised browser plugins, and using other techniques to gain remote access to vast numbers of addresses.

Today, malicious users are able to rotate IPs on a per-request basis.

Reblaze regularly sees attacks where each request comes in on a unique IP. (In larger incidents, we'll see 20-25K malicious requests per minute, rotating through IPs from all over the world, in continuous attacks that go on for several weeks. Ultimately, millions of addresses will be used.) In these situations, there's no single IP that can be quarantined. A specific IP will not be used more than once.

Therefore, IP and geolocation are no longer "facts" associated with attackers. They are not useful foundations for detecting and tracking web threats.

Does this imply that they are completely useless? No; many less-sophisticated attackers do not rotate IPs. Therefore, blacklists and other IP-based security policies can still be used as a low-overhead method to block them. But IP-based methods are only effective against a fraction of hostile Internet traffic today, and the percentage is shrinking rapidly.

The challenge of API protection

Traditional detection methods were originally developed for scrubbing website traffic. However, for many organizations today, mobile/native apps and microservices account for a significant portion of their traffic. Threat actors frequently reverse-engineer the underlying APIs, for example by downloading a mobile client from an app store and sniffing its communications with the endpoint. Hackers then program bots to mimic application behavior and attack vulnerable spots within it (e.g., credit card validation, credential stuffing, brute force logins, gift cards, coupon codes, and more). Any communication that can be initiated by a legitimate API user can also be abused by automated traffic from a botnet. For a large application, the potential damage can be millions of dollars per month.

Unfortunately, many of the traditional methods for bot detection are not useful for API protection. For example, an API user has no web browser environment that could be verified.

Conclusion

Older bots still make up a significant portion of automated traffic. And as mentioned above, traditional detection methods are still effective against them. Therefore, bot management solutions still use these methods, because a significant tranche of bots can be detected and blocked without large computational workloads.

Nevertheless, to detect more sophisticated bots—which are an increasingly large percentage of Internet traffic today, for both web applications and APIs—newer detection methods are needed.

What’s Working Today

In combination with the methods described above, modern bot detection solutions also incorporate newer techniques.

Traditional detection methods rely on many metrics which are passive (e.g., resource consumption rate), where the IDS waits to see what the requestor will do. A better approach is to proactively verify the authenticity and behavior of the requestor.

The techniques described below are not mutually exclusive. In any given situation, the IDS should use as many of them as possible. (Reblaze includes the methods described below, along with many other proprietary techniques not included here.)

Client Certification

Thanks to globally-available third-party server certificates, clients can be assured that when a connection is established, it has been made to the correct, legitimate endpoint. The next logical step is to provide mutual authentication by requiring that client certification be used as well.

This is common in some industries (e.g., FinTech), but is still rare in others. Reblaze provides a client certification mechanism within our platform, and we encourage our customers to use it.

Other Client Authentication

Along with client certificates, clients can be authenticated in other ways, e.g. native/mobile applications can have built-in methods of communicating with the endpoints and verifying their legitimacy.

At Reblaze, we provide an SDK (for both Android and iOS) to our customers, who rebuild and publish their applications with the SDK embedded. In use, it signs the application, authenticates the device, and verifies user identity. All communications occur over TLS and include an HMAC signature (a cryptographic identity mechanism on the client side) to harden communications between the mobile/native application and the microservice/API endpoint. The signatures are non-reproducible, non-guessable, non-repeating (they are unique per session, and sometimes, per request), and are based on dozens of parameters (time-based, location-based, environment-based, and more). They provide a reliable, secure mechanism to verify that the packets are originating from a legitimate user, and not from an emulator or other bot. Applications which use the Reblaze SDK are also authenticated by other methods, such as the Biometric Behavioral Profiling discussed in the next section.

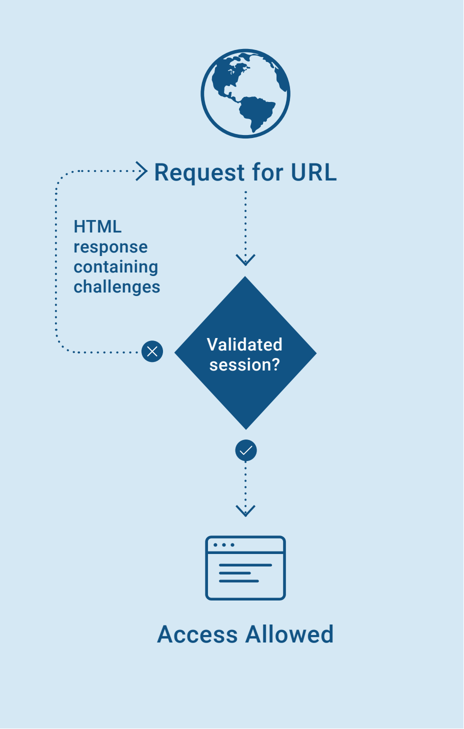
Advanced Browser Verification

Previously, we saw that traditional methods of verifying browser environments have become obsolete. For example, the first generation of headless browsers couldn't execute JavaScript. Today, most automated environments have the ability to do so. Therefore, simple JS verification is no longer effective.

As a result, some security solutions are now using more rigorous challenges for attempted usage of web applications. For example, a WAF can respond to an incoming request with a JS-based arithmetic challenge. If the client "browser" cannot solve the challenge correctly, it is deemed to be headless.

Other ways of testing browser capabilities can also be used; for example, the browser can be asked to render a sound or an image.

As part of its advanced environmental detection, Reblaze has developed many additional proprietary techniques. (Some are based on JS, while others are not.)



UEBA: User and Entity Behavioral Analytics

UEBA describes a process where data about user behavior is fed back into the intrusion detection process. An IDS uses UEBA to establish a baseline for legitimate user behavior, comparing subsequent users to this baseline in order to assess hostile intent.

UEBA is an approach, not a specific technique. Some security solutions are not yet using it at all; others have adopted it in basic form, while a few are applying it in sophisticated ways.

1. Basic UEBA

In its basic form, UEBA contrasts simple metrics from a current user to baseline values.

For example, a web application might require textual entries from users. Human typing tempos range from hunt-and-peck up to 150+ words per minute, generally at an irregular tempo. If a "user" enters long strings at a rapid, mechanically regular pace, or quickly enters strings into several different fields, then it's probably a bot.

Other interface metrics can be monitored as well. For example, if the same pixel is clicked in multiple checkboxes in rapid succession, and without corresponding mouse movements, this is probably not a human user.

2. Biometric Behavioral Profiling and Machine Learning (ML)

Many modern bots can mimic human behavior to varying degrees. Some are capable of evading detection by basic UEBA.

To defeat these bots, more advanced uses of UEBA are required.

One of the most effective approaches today is to use Machine Learning (ML) to construct behavioral profiles based upon biometrics: measurable characteristics of biological (i.e., human) users.

Conceptually, this is the same idea as basic UEBA: it compares the characteristics of current requestors to predetermined criteria which define 'good' users. In practice, it is much more complicated than the examples described earlier. For each protected API or application, comprehensive profiles of legitimate human behaviors are compiled. In this context, "comprehensive" refers to using all the data that is available.

As an illustration, here is how Reblaze does this. Every http/s request that Reblaze receives is anonymized and then analyzed according to numerous factors, including (partial list):

- **Device and software data**
  the user's hardware, its screen resolution and orientation, the software used, battery level, stack trace, fronts and extensions, emulator detection, window size, hidden iframes, etc.
- **User interface and events**
  mouse/pointer movements, clicks, taps, zooms, scrolls, keystrokes, speed of entry, etc.
- **Session data**
  requests sent, IPs used, timing, frequency, etc.
- **Consumption analytics**
  pages viewed, time spent, resources requested, etc.
- **Application-specific events**
  and other results of user actions.
- **And more.**

After an initial learning period, the platform understands the patterns, typical values, and common relationships of these metrics for legitimate users of each protected application and API.

The amount of data that Reblaze processes (over four billion requests per day) is far beyond the capability of human analysts. Therefore, cloud-based compute resources are used, applying ML in order to recognize patterns that analysts could not have identified on their own, or for which they might not have thought to look.

Unlike the basic UEBA described earlier, biometric behavioral profiling usually produces weighted variables. Instead of enforcing a list of static rules which modern bots can evade (e.g., "*if text entry rate > 150 wpm, deny requestor as bot*"), behavioral profiling monitors the combined relative weights of a list of behavioral metrics for each current requestor. If a requestor's combined 'score' ever goes above a preset threshold, that requestor is identified as a bot and blocked.

UEBA is meant to detect anomalous behavior of users and machines by comparing current metrics against a baseline of 'normal' behavioral events. The analysis is focused on finding anomalies and deviations from what has been established as normal and safe.

This approach is very different than the usual paradigm that older web security products were based upon (i.e., a WAF that enforces static rulesets). Thus, a number of security solutions still do not offer UEBA today.

3. Granular Profiling

Behavioral profiling based on Machine Learning can, and should, be performed as granularly as possible: not only per app, but even down to individual pages, screens, and so on. This can be extremely powerful.

For example: if a mobile/native app displays a map, and a high percentage of legitimate users zoom into it as their first action, then API users who do not submit zoom actions are more suspect. On a retail site, if legitimate visitors to a product page often scroll to a certain part of the page (perhaps to confirm that there's a money-back guarantee) before choosing "add to cart", then visitors who do not scroll are more suspect. And so on.

Of course, in these examples, not every legitimate user or visitor will perform the actions noted. That's why these are weighted factors, not binary decisions. But when enough "bot vs. human" indicators accumulate, high levels of accuracy can be achieved.

Using this approach, IDS accuracy is not only high, it is also robust and resistant to reverse-engineering by threat actors. Behavioral profiles are constructed based on private analytics data, and threat actors have no realistic way of obtaining this information.

Plus, unlike the basic UEBA examples discussed previously, many of the legitimate behavioral patterns will be non-obvious. ML often reveals useful patterns and relationships which few human analysts would have even considered.

Summary: Modern Bot Detection

As discussed previously, tracking a requestor based on traffic source (IP and geolocation) is no longer reliable. Today, each requestor's *identity* and *behavior* must be treated as fundamentals.

Behavior is especially important. After all, even users that are demonstrably human will not necessarily have benign intentions. But all hostile requestors (whether bot or human) must, at some point, deviate from legitimate user behavior. Once they do, behavioral profiling will identify them.

Frontiers of Bot Detection

As hostile bots continue to grow more sophisticated, security solutions must also evolve. Below we discuss some promising areas for further progress in bot detection: performance optimization, business outcome optimization, and new sources of data.

Optimizing Performance

Over the last few years, bot detection methods have evolved considerably.
The approaches used today can be categorized as follows:

1. **Static rules**
2. **Statistical analysis**
3. **Machine Learning (ML)**
4. **Combination of ML and statistical analysis**

This list represents a rough progression from less-effective approaches that are fast and inexpensive, to more-powerful analysis methods that are slower and resource-intensive.

For an IDS to have optimal performance in bot detection, it must apply the correct mix of these four approaches. It should use as much as possible of the earlier methods for fast detection, with sufficient use of the latter methods to ensure high accuracy.

1. Static Rules

The simplest form of detection relies on the enforcement of static rulesets. These can be based on ACLs (Access Control Lists), sizing limits, time limits, blacklisting of IPs, and so on.

This approach requires minimal processing, and thus it is fast and cheap. However, static rules can be evaded fairly easily by the latest bots. As discussed previously, bots can change IP addresses, reduce the speed at which they submit requests, and so on.

An additional disadvantage is that overreliance on static rule enforcement will tend to increase the rate of FP (False Positive) alarms. As rulesets become more stringent, legitimate users will begin to violate them inadvertently. For example, a reduction in the allowable number of requests per session will flag users with longer sessions, and they will be incorrectly identified and blocked as bots.

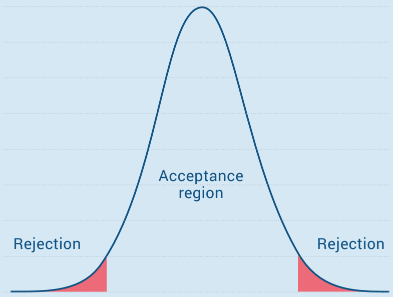
Nevertheless, static rules still play a useful role in bot detection. Moderately-strict rulesets can still detect a large percentage of hostile bots with a very low rate of FPs. And there's no reason to expend excessive computing workloads on catching these bots when it's still possible to detect them using low-overhead methods.

2. Statistical Approaches

Statistical analysis of incoming traffic requires more processing than static rule enforcement, but it is still straightforward to implement.

Applying widespread statistical tests such as Student's t-test, Z-test, and analysis of variance (ANOVA) can often give transparent results. The base of these methods is a calculation of standard deviations and determination of borders between normal and abnormal numbers of chosen metrics.

These analyses are usually run on aggregated features or raw parameters extracted from traffic logs. Unlike static rule enforcement, statistical traffic analysis does not occur in real time. However, statistical analysis can be used to create 'local' or custom rules, which are then enforced in real time along with static rulesets. An ideal IDS will have minimal lag between the receipt of each request and its subsequent analysis, and it will frequently update its 'local' rules as needed.



Statistical methods such as the Z-test provide clear, well-understood analytical methods to determine anomalies within a dataset.

3. Machine Learning

Although applying ML to bot detection is not a new idea, many security solutions still are not doing this. Nevertheless, it is a necessity today due to the increasing sophistication of hostile bots. Today's 'smart' attackers adapt to static limitations, minimize statistical anomalies, and otherwise program their bots to avoid detection.

Earlier ML algorithms were based on a search for anomalies and baseline plotting. The conventional method for outliers detection is RANSAC (i.e., "random sample consensus." RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates). The general criteria for dataset separation were abnormal and normal behaviors.

Some Machine Learning algorithms can see hidden interconnections (nonobvious features) in the data. In the case of unsupervised techniques, they can help to do a customized selection of significant features connected to dataset-specific characteristics.

This makes ML more effective than statistical methods for detecting sophisticated bots which are masquerading as human users.

Today, anomalies have much more complex foundations. Bot attacks are difficult to detect because many bots have adopted algorithms to imitate human behavior. In other words, an IDS needs to detect bots which mask themselves as legitimate users.

As with statistical approaches, ML analysis is performed on aggregated results and then translated into rules for one-step usage in real time. However, ML requires larger datasets; every extracted feature (x) requires 10^X observations. This makes ML a bit less flexible than purely statistical approaches. ML is also significantly more technical; it can be difficult for users to make optimal choices for feature significance, data distribution, and so on.

4. Combining ML and Statistical Techniques

When choosing an approach for bot detection, it's important to understand that a perfect, universally applicable method cannot exist. Even if theoretical results would give a perfect correlation with real-life datasets, threat actors would develop a new generation of bots which would leverage techniques to circumvent any static rules. (Indeed, as the next section will discuss, we shouldn't even expect a universally applicable standard for assessing bot detection methods.)

In general, a combination of approaches such as static rules, statistical analysis, and ML will be the best solution for real-world applications. Some bots can still be blocked with fast, inexpensive methods. More sophisticated bots can then be detected with more intensive analysis.

The challenge for IDS providers is to build this flexibility into their platforms with an accessible interface that doesn't require too much ML expertise from users.

Optimizing for Business Outcomes

When evaluating a bot detection solution, what metric(s) should be used?

Accuracy is the most intuitive performance measure, and it is straightforward to calculate. It is the ratio of correctly predicted observations to the total observations. But there's more to consider than this.

It is tempting to seek an "ultimate" bot detection algorithm that is the most accurate of all possible algorithms, with the intent of applying it across all possible situations. However, such an algorithm does not exist. Different use cases are best addressed with different approaches, and customizing an algorithm for one use case makes it less effective in others.

Further, accuracy as described above is an incomplete metric. False Positive (FP) and False Negative (FN) events must also be considered.

When an FP alarm occurs, a legitimate user request is misidentified by the IDS as being hostile, and is therefore blocked. An FN is the inverse of this: a malicious request is incorrectly identified as being legitimate, and it is allowed through.

Thus, a complete measurement of accuracy becomes:

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)}$$

(Where TP and TN represent True Positive and True Negative events, respectively.)

However, even this improved metric is not necessarily enough to produce optimal business outcomes. One might think that a higher level of accuracy is always better, but this is not always the case.

FP and FN events are both undesirable, and an IDS should minimize them both. However, they are not necessarily equal; one can have much harsher consequences than the other.

For example, in retail, FPs can create direct losses of revenue when customers are prevented from buying. Meanwhile, for other organizations, FNs might have worse consequences. For example, if they facilitate the exposure of sensitive customer data from an organization in a highly-regulated industry, this can result in punitive fines.

Therefore, when individual organizations fine-tune their IDS performance, they must consider more than just overall accuracy.

For example, it's possible that the highest-possible accuracy is accompanied with minimal FPs but a higher level of FNs, as compared to a slightly lower accuracy which includes higher FPs but lower FNs. In an industry where FNs are a serious matter, the slightly lower accuracy might be more desirable.

We see then that for an IDS to produce optimal results for an organization, it should be customized for each use case. The definition of "optimal" will vary from organization to organization, and indeed, even within the same organization it can vary across web applications.

Historically, the ability to customize an IDS to this level has been rare. For a web security solution to offer full customization, it must expose a variety of algorithmic parameters to users, while providing in-depth feedback about the resulting performance changes, and do it all in an understandable UI that doesn't overwhelm users with details.

This task is obviously not easy. But as bot detection becomes more technically challenging, and as the relative impacts of FPs and FNs become potentially more variable (with the still-rising commercial importance of the web, an increasingly restrictive regulatory environment surrounding consumer privacy, etc.), web security solution providers will need to add this type of customization to their products.

Of course, this raises an obvious question: how can IDS users measure the FP and/or FN rates? By definition, the IDS itself cannot provide this information. This question brings us to our next topic.

Leveraging New Sources of Data

Traditional bot detection relies on data captured from the incoming traffic stream. However, this is not the only potential source of useful information.

Conversion rate (CR) is a rich source of insights into the success of a web application. It is the ratio of conversions to visitors or requests, where a "conversion" represents the attainment of a desired goal (e.g., a purchase, registration, download, etc.)

Conversions occur when an incoming request corresponds to a specified 'target' action within the web application (clicking on a "checkout" button, submitting a filled-in lead-generation form, and so on). Each time a target action occurs, the application's conversion rate is positively affected.

Malicious bots usually do not perform typical target actions. Therefore, when an IDS correctly identifies and blocks bots, the web application's conversions will not go down.

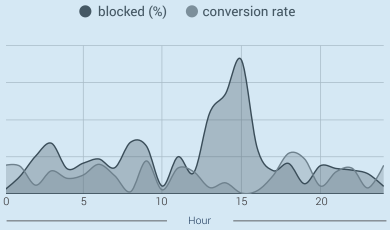
However, when an IDS generates FP alarms, then some legitimate users are being filtered as bots, and they are prevented from taking actions they would otherwise have taken. This decreases the CR.

This provides an opportunity to use application CR for feedback and algorithmic improvement. In other words, as potential "improvements" to the IDS are made (via adjustments to its algorithms, parameters, and so on), the CR can be monitored. If the adjustments caused the CR to decrease, then it's quite possible that the FP rate increased (as shown in the top example to the right). Further investigation should be done immediately.

On the other hand, if IDS adjustments resulted in a higher percentage of requests being blocked, but the CR was not affected, then it would seem that the rate of false alarms has not changed, and the IDS adjustments were successful. (See bottom example to the right.)

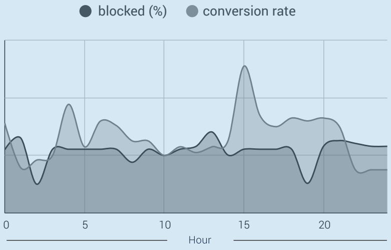
A brief aside: The discussion above assumes that conversion rate is calculated based upon all incoming requests, before the IDS determines which "visitors" are bots. If instead CR is calculated using legitimate visitors (i.e., the visitors remaining after the removal of requestors that were identified as bots), then the relationships described above become somewhat more complicated. For example, a change in the rate of FNs will not affect CR under the "all requests" approach, but it will affect CR under the "only legitimate requests" approach. There are other complicating factors as well. In any case, the overall point remains: CR is a useful metric for improving the effectiveness of an IDS.

Blocked requests vs. Conversion Rate



Comparison of blocked requests to conversion rate. For hour 15, as more requests were blocked, conversion rate fell to near-zero. This indicates a probable high rate of FPs being generated by the IDS.

Blocked requests vs. Conversion Rate



Comparison of blocked requests to conversion rate. There appears to be little correlation between blocked/passed requests and CR. The fluctuations in CR are thus attributable to other factors (e.g., normal variations in buyer behavior throughout the day), rather than being the result of FPs and FNs.

Summary: network traffic logs are no longer the only source of data to analyze. The accuracy of an IDS' bot detection can also be improved by integrating its UEBA with analytical platforms such as Google Analytics.

Furthermore, this can provide insights into the rates of FP and FN generation.

Conclusion

Effective bot protection is crucial for any organization with significant web assets. On average, almost 40 percent of all incoming traffic consists of malicious bots. The predominant types vary across verticals, but they all can result in harmful consequences for the targeted sites and applications.

Unfortunately, most IDS solutions today are still relying on older detection methods such as blacklisting, rate limiting, signature detection, and reCAPTCHAs. These are growing increasingly incapable of identifying today's hostile bots.

Modern approaches such as ML-based UEBA and granular Biometric Behavioral Profiling are essential tools for reliable bot control today. These technologies are not yet commonplace among security solutions, but reliable detection of malicious bots requires them.

For the future, as mentioned in the Frontiers section, there are many promising ways in which bot detection can evolve beyond its current forms. Reblaze is actively integrating these capabilities into its platform already today, while pushing forward into new areas of research.

Questions about the content of this white paper? Contact us at hello@reblaze.com.

To receive notifications of future publications, sign up for our newsletter by filling out the form at www.reblaze.com/contact-us/.

Reblaze Technologies, Ltd.
3031 Tisch Way
110 Plaza West
San Jose, CA 95128