API v2

The API allows the list of pwned accounts (email addresses and usernames) to be quickly searched via a RESTful service. Check out who's currently using the API (/API/Consumers).

Overview

You're reading about v2 of the API which is presently the current version and returns a significant amount of additional data over v1. You can still access the documentation for v1 (/API/v1) which continues to be available in the form it was originally released in.

Index

Specifying the API version

The second version of the API is consumable in a variety of different fashions depending on how you'd like to specify the version. For more background, read Your API versioning is wrong, which is why I decided to do it 3 different wrong ways (https://www.troyhunt.com/your-api-versioning-is-wrong-which-is/). Regardless of the implementation you choose, they will all continue to be supported.

Via the URL

This method can easily be invoked directly by requesting the URL and does not require any request header manipulation. As such, you can test this API directly in the browser, for example by clicking here (/api/v2/breachedaccount/test@example.com).

```
GET https://haveibeenpwned.com/api/v2/{service}/{parameter}
```

Via an api-version header

This method provides a stable URL depicting the resource being requested and will not change over versions of the API. The version is specified by adding a custom request header called "api-version".

```
GET https://haveibeenpwned.com/api/{service}/{parameter}
api-version: 2
```

Via content negotiation

This method allows the version to be specified using content negotiation (https://en.wikipedia.org/wiki/Content_negotiation). The version is passed in the accept header using the "application/vnd.haveibeenpwned.v{version}+json" pattern.

```
GET https://haveibeenpwned.com/api/{service}/{parameter}
Accept: application/vnd.haveibeenpwned.v2+json
```

Specifying the user agent

Each request to the API must be accompanied by a user agent request header. Typically this should be the name of the app consuming the service, for example "Pwnage-Checker-For-iOS". A missing user agent will result in an HTTP 403 response. A valid request would look like:

```
GET https://haveibeenpwned.com/api/{service}/{parameter}
User-Agent: Pwnage-Checker-For-iOS
```

The user agent should accurately describe the nature of the API consumer such that it can be clearly identified in the request.

Getting all breaches for an account

The most common use of the API is to return a list of all breaches a particular account has been involved in. The API takes a single parameter which is the account to be searched for. The account is not case sensitive and will be trimmed of leading or trailing white spaces. The account should always be URL encoded.

By version in URL (testable by clicking here (/api/v2/breachedaccount/test@example.com)):

```
GET https://haveibeenpwned.com/api/v2/breachedaccount/{account}
```

By api-version header:

```
GET https://haveibeenpwned.com/api/breachedaccount/{account}
api-version: 2
```

By content negotiation:

```
GET https://haveibeenpwned.com/api/breachedaccount/{account}
Accept: application/vnd.haveibeenpwned.v2+json
```

If the complete breach data is not required and you'd like to reduce the response body size, you can request that the breach entity be truncated so that only the name attribute is returned (reduces response body size by approximately 98%):

Parameter	Example	Test	Description
truncateResponse	?truncateResponse=true	test (/api/v2/breachedaccount/test@example.com?truncateResponse=true)	Returns only the name of the breach.

The result set can also be filtered by passing one of the following query strings:

Parameter	Example	Test	Description
domain	?domain=adobe.com	test (/api/v2/breachedaccount/test@example.com?domain=adobe.com)	Filters the result set to only breaches against the domain specified. It is possible that one site (and consequently domain), is compromised on multiple occasions.

Note: the public API *will not* return accounts from any breaches flagged as sensitive (/FAQs#SensitiveBreach) or retired (/FAQs#RetiredBreach). By default, the API also won't return breaches flagged as unverified (/FAQs#UnverifiedBreach), however these can be included by using the following parameter:

Parameter	Example	Test	Description
includeUnverified	?includeUnverified=true	test (/api/v2/breachedaccount/test@example.com?includeUnverified=true)	Returns breaches that have been flagged as "unverified". By default, only verified breaches are returned web performing a search.

Getting all breached sites in the system

A "breach" is an instance of a system having been compromised by an attacker and the data disclosed. For example, Adobe was a breach, Gawker was a breach etc. It is possible to return the details of each of breach in the system which currently stands at **293 breaches**.

By version in URL (testable by clicking here (/api/v2/breaches)):

```
GET https://haveibeenpwned.com/api/v2/breaches
```

By api-version header:

```
GET https://haveibeenpwned.com/api/breaches
api-version: 2
```

By content negotiation:

```
GET https://haveibeenpwned.com/api/breaches
Accept: application/vnd.haveibeenpwned.v2+json
```

The result set can also be filtered by passing one of the following query strings:

Parameter	Example	Test	Description

Parameter	Example	Test	Description
domain	?domain=adobe.com	test (/api/v2/breaches?domain=adobe.com)	Filters the result set to only breaches against the domain specified. It is possible that one site (and consequently domain), is compromised on multiple occasions.

Getting a single breached site

Sometimes just a single breach is required and this can be retrieved by the breach "name". This is the stable value which may or may not be the same as the breach "title" (which can change). See the breach model below for more info.

By version in URL (testable by clicking here (/api/v2/breach/Adobe)):

```
GET https://haveibeenpwned.com/api/v2/breach/{name}
```

By api-version header:

```
GET https://haveibeenpwned.com/api/breach/{name}
api-version: 2
```

By content negotiation:

```
GET https://haveibeenpwned.com/api/breach/{name}
Accept: application/vnd.haveibeenpwned.v2+json
```

Getting all data classes in the system

A "data class" is an attribute of a record compromised in a breach. For example, many breaches expose data classes such as "Email addresses" and "Passwords". The values returned by this service are ordered alphabetically in a string array and will expand over time as new breaches expose previously unseen classes of data.

By version in URL (testable by clicking here (/api/v2/dataclasses)):

```
GET https://haveibeenpwned.com/api/v2/dataclasses
```

By api-version header:

```
GET https://haveibeenpwned.com/api/dataclasses
api-version: 2
```

By content negotiation:

```
GET https://haveibeenpwned.com/api/dataclasses
Accept: application/vnd.haveibeenpwned.v2+json
```

The breach model

Each breach contains a number of attributes describing the incident. In the future, these attributes may expand *without* the API being versioned. The current attributes are:

Attribute	Type	Description
Name	string	A Pascal-cased name representing the breach which is unique across all other breaches. This value never changes and may be used to name dependent assets (such as images) but should not be shown directly to end users (see the "Title" attribute instead).
Title	string	A descriptive title for the breach suitable for displaying to end users. It's unique across all breaches but individual values may change in the future (i.e. if another breach occurs against an organisation already in the system). If a stable value is required to reference the breach, refer to the "Name" attribute instead.
Domain	string	The domain of the primary website the breach occurred on. This may be used for identifying other assets external systems may have for the site.
BreachDate	date	The date (with no time) the breach originally occurred on in ISO 8601 format. This is not always accurate — frequently breaches are discovered and reported long after the original incident. Use this attribute as a guide only.
AddedDate	datetime	The date and time (precision to the minute) the breach was added to the system in ISO 8601 format.
ModifiedDate	datetime	The date and time (precision to the minute) the breach was *modified* in ISO 8601 format. This will only differ from the AddedDate attribute if other attributes represented here are changed or data in the breach itself is changed (i.e. additional data is identified and loaded). It is always either equal to or greater then the AddedDate attribute, never less than.
PwnCount	integer	The total number of accounts loaded into the system. This is usually less than the total number reported by the media due to duplication or other data integrity issues in the source data.
Description	string	Contains an overview of the breach represented in HTML markup. The description may include markup such as emphasis and strong tags as well as hyperlinks.

Attribute	Type	Description
DataClasses	string[]	This attribute describes the nature of the data compromised in the breach and contains an alphabetically ordered string array of impacted data classes.
IsVerified	boolean	Indicates that the breach is considered unverified (/FAQs#UnverifiedBreach). An unverified breach *may not* have been hacked from the indicated website. An unverified breach is still loaded into HIBP when there's sufficient confidence that a significant portion of the data is legitimate.
IsFabricated	boolean	Indicates that the breach is considered fabricated (/FAQs#FabricatedBreach). A fabricated breach is *unlikely* to have been hacked from the indicated website and usually contains a large amount of manufactured data. However, it still contains legitimate email addresses and asserts that the account owners were compromised in the alleged breach.
IsSensitive	boolean	Indicates if the breach is considered sensitive (/FAQs#SensitiveBreach). The public API *will not* return any accounts for a breach flagged as sensitive.
IsRetired	boolean	Indicates if the breach has been retired (/FAQs#RetiredBreach). This data has been permanently removed and will not be returned by the API.
IsSpamList	boolean	Indicates if the breach is considered a spam list (/FAQs#SpamList). This flag has no impact on any other attributes but it means that the data has not come as a result of a security compromise.

Sample breach response

All responses returns breach models either in a collection (breaches for account or all breaches in the system) or as a single item (retrieving a breach by name). When a collection is returned, it's sorted alphabetically by the *title* of the breach.


```
[
{
"Name":"Adobe",
"Title":"Adobe",
"Domain":"adobe.com",
"BreachDate":"2013-10-04",
"AddedDate":"2013-12-04T00:00Z",
"ModifiedDate":"2013-12-04T00:00Z",
"PwnCount":152445165,
"Description":"In October 2013, 153 million Adobe accounts were breached
with each containing an internal ID, username, email, <em>encrypted</em>
password and a password hint in plain text. The password cryptography
was poorly done and <a href=\"http://stricture-group.com/files/adobe-
```

```
top100.txt\" target=\"_blank\" rel=\"noopener\">many were quickly
resolved back to plain text</a>. The unencrypted hints also <a
href=\"http://www.troyhunt.com/2013/11/adobe-credentials-and-
serious.html\" target=\"_blank\" rel=\"noopener\">disclosed much about
the passwords</a> adding further to the risk that hundreds of millions
of Adobe customers already faced.",
"DataClasses":["Email addresses","Password
hints","Passwords","Usernames"],
"IsVerified":True,
"IsSensitive":False,
"IsRetired":False,
"IsSpamList":False
},
{
"Name":"BattlefieldHeroes",
"Title":"Battlefield Heroes",
"Domain":"battlefieldheroes.com",
"BreachDate":"2011-06-26",
"AddedDate":"2014-01-23T13:10Z",
"ModifiedDate":"2014-01-23T13:10Z",
"PwnCount":530270,
"Description":"In June 2011 as part of a final breached data dump, the
hacker collective &amp;quot;LulzSec&amp;quot; <a
href=\"http://www.rockpapershotgun.com/2011/06/26/lulzsec-over-release-
battlefield-heroes-data\" target=\"_blank\" rel=\"noopener\">obtained
and released over half a million usernames and passwords from the game
Battlefield Heroes</a>. The passwords were stored as MD5 hashes with no
salt and many were easily converted back to their plain text versions.",
"DataClasses":["Passwords","Usernames"],
"IsVerified":True,
"IsSensitive":False,
"IsRetired":False,
"IsSpamList":False
}
]
```

Getting all pastes for an account

The API takes a single parameter which is the email address to be searched for. Unlike searching for breaches, usernames that are not email addresses cannot be searched for. The email is not case sensitive and will be trimmed of leading or trailing white spaces. The email should always be URL encoded.

By version in URL (testable by clicking here (/api/v2/pasteaccount/test@example.com)):

```
GET https://haveibeenpwned.com/api/v2/pasteaccount/{account}
```

By api-version header:

```
GET https://haveibeenpwned.com/api/pasteaccount/{account}
api-version: 2
```

By content negotiation:

```
GET https://haveibeenpwned.com/api/pasteaccount/{account}
Accept: application/vnd.haveibeenpwned.v2+json
```

The paste model

Each paste contains a number of attributes describing it. In the future, these attributes may expand *without* the API being versioned. The current attributes are:

Attribute	Type	Description
Source	string	The paste service the record was retrieved from. Current values are: Pastebin, Pastie, Slexy, Ghostbin, QuickLeak, JustPaste, AdHocUrl, OptOut
Id	string	The ID of the paste as it was given at the source service. Combined with the "Source" attribute, this can be used to resolve the URL of the paste.
Title	string	The title of the paste as observed on the source site. This may be null and if so will be omitted from the response.
Date	date	The date and time (precision to the second) that the paste was posted. This is taken directly from the paste site *when this information is available* but may be null if no date is published.
EmailCount	integer	The number of emails that were found when processing the paste. Emails are extracted by using the regular expression \b+(?!^.{256})[a-zA-Z0-9\.\-_\+]+@[a-zA-Z0-9\.\-_]+\.[a-zA-Z]+\b

---

Sample paste response

Searching an account for pastes always returns a collection of the paste entity. The collection is sorted chronologically with the *newest* paste first.

```
[
{
"Source":"Pastebin",
"Id":"8Q0BvKD8",
"Title":"syslog",
"Date":"2014-03-04T19:14:54Z",
"EmailCount":139
},
{
"Source":"Pastie",
"Id":"7152479",
"Date":"2013-03-28T16:51:10Z",
"EmailCount":30
}
]
```

Response codes

Semantic HTTP response code are used to indicate the result of the search:

Code	Description
200	Ok — everything worked and there's a string array of pwned sites for the account
400	Bad request — the account does not comply with an acceptable format (i.e. it's an empty string)
403	Forbidden — no user agent has been specified in the request
404	Not found — the account could not be found and has therefore not been pwned
429	Too many requests — the rate limit has been exceeded

Pwned Passwords overview

Pwned Passwords are more than half a billion passwords which have previously been exposed in data breaches. The service is detailed in the launch blog post (https://www.troyhunt.com/introducing-306-million-freely-downloadable-pwned-passwords/) then further expanded on with the release of version 2 (https://www.troyhunt.com/ive-just-launched-pwned-passwords-version-2). The entire data set is both downloadable and searchable online via the Pwned Passwords page (/Passwords). It's also queryable via the following two APIs:

Each password is stored as a SHA-1 hash of a UTF-8 encoded password. The downloadable source data delimits the full SHA-1 hash and the password count with a colon (:) and each line with a CRLF.

Searching by range

In order to protect the value of the source password being searched for, Pwned Passwords also implements a k-Anonymity model (https://en.wikipedia.org/wiki/K-anonymity) that allows a password to be searched for by partial hash. This allows the first 5 characters of a SHA-1 password hash (not case-sensitive) to be passed to the API (testable by clicking here (https://api.pwnedpasswords.com/range/21BD1)):

```
GET https://api.pwnedpasswords.com/range/{first 5 hash chars}
```

When a password hash with the same first 5 characters is found in the Pwned Passwords repository, the API will respond with an HTTP 200 and include the *suffix* of every hash beginning with the specified prefix, followed by a count of how many times it appears in the data set. The API consumer can then search the results of the response for the presence of their source hash and if not found, the password does not exist in the data set. A sample response for the hash prefix "21BD1" would be as follows:

```
0018A45C4D1DEF81644B54AB7F969B88D65:1
00D4F6E8FA6EECAD2A3AA415EEC418D38EC:2
011053FD0102E94D6AE2F8B83D76FAF94F6:1
012A7CA357541F0AC487871FEEC1891C49C:2
0136E006E24E7D152139815FB0FC6A50B15:2
...
```

On average, a range search returns 478 hash suffixes, although this number will differ depending on the hash prefix being searched for. The smallest result is 381, the largest 584. There are 1,048,576 different hash prefixes between 00000 and FFFFF (16^5) and every single one will return HTTP 200; there is no circumstance in which the API should return HTTP 404.

Code	Body	Description
200	Hash suffixes counts	Ok — all password hashes beginning with the searched prefix are returned alongside prevalence counts

Read more about how k-Anonymity and the Pwned Passwords range search protects searched passwords. (https://www.troyhunt.com/ive-just-launched-pwned-passwords-version-2)

HTTPS

All API endpoints must be invoked over HTTPS. Any requests over HTTP will result in a 301 response with a redirect to the same path on the secure scheme. Only TLS versions 1.2 and 1.3 are supported; older versions of the protocol will not allow a connection to be made.

Cross-origin resource sharing (CORS)

CORS (https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) is fully supported for all origins — you can hit the API from websites on any other domain.

Authentication

There isn't any.

Rate limiting

Requests to the breaches and pastes APIs are limited to one per every 1500 milliseconds each from any given IP address (an address may request both APIs within this period). Any request that exceeds the limit will receive an HTTP 429 "Too many requests" (https://httpstatuses.com/429) response. The response also includes an accompanying "Retry-After" response header expressing the number of seconds remaining before the IP address can make a successful API call (the value is rounded up to the next whole second). The response body explains the rate limit and refers to the acceptable use documentation.

A typical response looks like this:

```
HTTP/1.1 429
Retry-After: 2
Rate limit exceeded, refer to acceptable use of the API:
https://haveibeenpwned.com/API/v2#AcceptableUse
```

The retry period is sliding; attempting to query the API more aggressively than the rate allows causes the retry period to start again with each failed request. It's advisable to avoid querying the API at exactly the rate limit as network behaviour may result in some requests arriving within the retry period and causing a 429. Adding a 100 millisecond delay between requests will usually ensure this won't happen.

Where the rate limit is consistently exceeded, further defences may be employed to limit the ability to query the API. These defences include blocks or JavaScript challenges by Cloudflare which may result in an HTTP 503 "Service Unavailable" response.

There is no rate limit on the Pwned Passwords API.

Abuse

There's not much point; if you want to build up a treasure trove of pwned email addresses or usernames, go and download the dumps (they're usually just a Google search away) and save yourself the hassle and time of trying to enumerate an API one account at a time. That said, use of the API should fall within acceptable use expectations:

Acceptable use

The API has been expressly designed to be *easy*; no auth and nothing to get in the way of people doing *awesome* things with it. Things that are *not* awesome include:

- Querying the data for purposes that are intended to cause harm to the victims of data breaches
- Prolonged and aggressive querying of the service such that it impacts availability *or* my costs
- Anything deliberately intended to limit service availability such as denial of service attacks
- Deliberate attempts to circumvent the rate limit or abuse other measures design to ensure acceptable use
- Not properly identifying the user agent such that it accurately describes the consumer of the API
- Misrepresenting the consuming client by impersonating other user agents in an attempt to obfuscate API requests
- Other services designed to fraudulently represent the Have I Been Pwned name or brand
- Misrepresenting the source of the data as originating from somewhere other than Have I Been Pwned
- Not adhering to the Creative Commons Attribution License as described below
- Using the service in a fashion that brings Have I Been Pwned into disrepute

Abusing these objectives may limit your ability to query the service via a range of countermeasures. If in doubt, get in touch (https://www.troyhunt.com/contact/) and outline how you'd like to use the service in a way that's consistent with these objectives.

License — breach & paste APIs

In other words, you're welcome to use the public API to build other services, but *you must identify Have I Been Pwned as the source of the data* . Clear and visible attribution with a link to haveibeenpwned.com (https://haveibeenpwned.com/) should be present anywhere data from the service is used including when searching breaches or pastes and when representing breach descriptions. It doesn't have to be overt, but the interface in which Have I Been Pwned data is represented should clearly attribute the source per the Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

In order to help maximise adoption, there is no licencing or attribution requirements on the Pwned Passwords API, although it is welcomed if you would like to include it.

A troyhunt.com project (https://www.troyhunt.com)

 (https://www.facebook.com/troyahunt) 

(https://twitter.com/troyhunt)  (https://www.troyhunt.com/contact/)

 (https://plus.google.com/+TroyHunt)