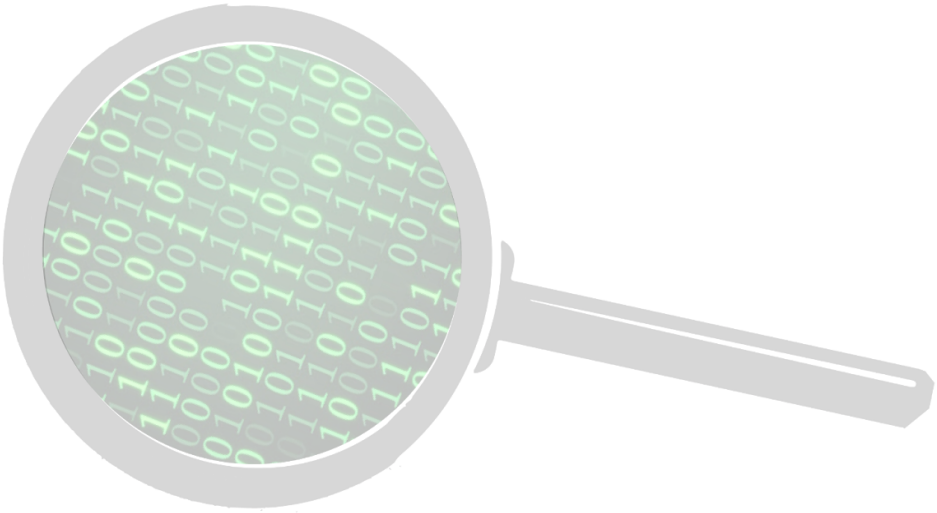
myNetWatchman

CompEmail API User Guide

Version 1.0



Contents

Introduction

Welcome to the reference documentation for the myNetWatchman (mNW) Restful API.

All API requests must be made over HTTPS. Requests made over plain HTTP will fail.

Our API accepts JSON-encoded requests and produces JSON-encoded results, unless otherwise noted. Be aware that myNetWatchman may add additional fields into JSON objects at any time.

Endpoints

You can access the myNetWatchman Restful APIs at the following endpoints:

Description	Base URL for REST Endpoints
Development	https://dev-api.mynetwatchman.com
Production	https://api.mynetwatchman.com

Compromised Email Overview

The Compromised Email APIs are intended to help you identify email accounts that have been compromised at some point in time. All of the email boxes in mNW's Compromised Email data set have been successfully accessed (with the correct password) by criminals. Users with compromised email accounts are a higher risk for various types of fraud. To help you determine risk, the API response includes the last time that mNW observed the unauthorized access to the email account. For example, the longer in the past of the unauthorized access, the higher the likelihood the legitimate user has changed the password and resolved the security situation.

Email Normalization Process

Normalization is the process of reducing an email address down to its most basic form so that meaningful comparisons can be made. For example, Gmail strips any dot ('.') characters in the local part of an email and anything in the local part beginning with a plus ('+') character is considered an alias, so "test.user+this@gmail.com" is normalized (and equivalent) to "testuser@gmail.com".

All of our compromised email records are stored in a normalized form so that thorough comparisons can be made. If you want to get the most value out of compromised email searches you will want to normalize the email address first, or call the API with the raw address and have the API do the normalization.

The normalization process may require a DNS lookup for an MX record, so if your application is timing critical you might want to search for the email address as-is first, and then normalize the email address out-of-band and run another search if it differs from the original.

Here is the normalization process used:

- Remove leading and trailing whitespace from the address
- Lower case the email address
- Remove trailing dots from the domain
- Obtain MX records for the (IDNA encoded) domain
- Compare MX records against provider specific rules and apply them if there is a match

We provide an API here that can be used to normalize email addresses. If you choose to implement this yourself, this can also be used as a comparison point to validate that our solutions match.

Here is a list of provider specific rules that we are using at this time. This list may expand as we discover more. If you know of additional provider specific rules not listed here, please submit them through our support channels.

Rule Types

- Plus Sub-addressing - Remove '+' style sub-addressing, e.g., john+doe@example.com = john@example.com
- Strip Dots - Remove '.' characters from an address, e.g., john.doe@example.com = johndoe@example.com
- Subdomain Addressing - Subdomain becomes the address, e.g., anything@john.example.com = john@example.com

Provider Specific Rules

Provider	MX Record Suffix(es)	Rules
Apple	.icloud.com.	Plus sub-addressing
Fastmail	.messagingengine.com.	Subdomain Addressing, Plus Sub-addressing
Google	.gmail.com. .googlemail.com. .google.com.	Strip Dots, Plus sub-addressing
MeMail	.memail.com.	Plus sub-addressing
Microsoft	.outlook.com.	Plus sub-addressing
Pobox	.pobox.com.	Plus sub-addressing
postale.io	.postale.io.	Plus sub-addressing
ProtonMail	.protonmail.ch.	Plus sub-addressing
Rackspace	.emailsrvr.com.	Plus sub-addressing
Runbox	.runbox.com.	Plus sub-addressing
Yandex	.yandex.ru.	Plus sub-addressing
Zoho	.zoho.com.	Plus sub-addressing

Compromised Email Testing

Test Data

In our development environment there is some synthetic test data and some real world data for testing.

Synthetic Data

There are three types of synthetic data represented:

1. Usernames that are guaranteed to exist with older last seen dates
2. Usernames that are guaranteed to exist with a last seen date in the last month
3. Usernames that are guaranteed not to exist

Usernames with old last seen dates follow the pattern `test_user_old_XXX@example.com` where `XXX` is a number between 000 and 999 inclusive. These will have a random discover date between 2018-01-01 and 2019-01-01.

Usernames with random last seen dates within the last month follow the pattern `test_user_XXX@example.com` where `XXX` is a number between 000 and 999 inclusive.

Usernames with the pattern `test_not_user_XXX@example.com` are guaranteed to not exist in the development environment.

Real-world data

In our development instance there is also data from 256,073 unique email accounts that we detected as compromised on June 23, 2021.

Testing in Production

If you run tests against our production instance, you should add the header `X-Test` to the requests. This will ensure that the metrics provided to you will not be tainted with test data. The synthetic test data is not available in our production environment.

Compromised Email API

This API checks if an email has been or is compromised.

In order to check if an email has been compromised, it should be normalized first.

The normalization process may require a DNS lookup for an MX record, so if your application is timing critical you may want to check the raw email address first (as a normalized address), and then normalize the email address out of band, and resubmit if the normalization process changes the address.

Types of queries:

- Check SHA256 hash of normalized email address
- Check SHA256 hash prefix of a normalized email address
- Check raw email address
- Check normalized email address

Test Endpoint

This API call will test access and connectivity to the Compromised Email API. The call will return a 200 HTTP code, and the text "pong" on success.

AUTHORIZATIONS: APIKeyAuth

API Key: ApiKeyAuth

When access is granted to the APIs, you are given one or more secret API keys. Each key will be designated for either the development environment or the production environment. A development key has the prefix `mnw_dev_` and a production key has the prefix `mnw_prod_`. A key from one environment will not work in the other environment. A production API key will be limited to APIs designated in your contract. The development API key will allow access to all APIs.

These keys are only for requests made from the server side. Never share these keys. Keep them guarded and secure. Publicly exposing your API keys could cause your account to be

compromised, which can result in unexpected charges. It is recommended that you further secure your API keys by providing myNetWatchman a white-list of source IP addresses or networks that your requests will originate from.

You authenticate to the myNetWatchman API by providing your secret key in the HTTP header `X-Api-Key`. Below is an example with curl.

```
curl -H 'x-api-key: [PROVIDE YOUR KEY HERE]'
'https://dev-api.mynetwatchman.com/allcreds/ping'
```

Example

GET /compemail/ping

```
curl
'https://dev-api.mynetwatchman.com/compemai
l/ping' \
  -H 'x-api-key: [PROVIDE YOUR KEY HERE]'
```

Responses

√ **200** Successful

```
pong
```

RESPONSE SCHEMA: text/plain

string

- 401 The `X-API-KEY` header was not present or empty

- 403 API key was not correct, the source IP is not in the white-list, or the API key is not entitled to this API

- 405 Incorrect method was used

Search Compromised Email Data

POST /compemail/search

AUTHORIZATIONS: APIKeyAuth

REQUEST BODY SCHEMA: application/json

- Echo_search
  - boolean
  - Set to true to include the search criteria in each result. Defaults to false

- Search
  - Array of sha256 (object) or norm (object) or raw (object)
  - Search criteria
  - Array [ ] Any of
    - sha256
      - format
        - string
        - Value: `"sha256"`
        - Type of search
      - value
        - string <sha256 hash> [5 .. 64] characters ^[0-9a-fA-F]{6,64}$
    - norm
      - format
        - string
        - Value: `"norm"`
        - Type of search
      - value
        - string <email>
        - Normalized email address
    - Raw
      - format
        - string
        - Value: `"raw"`
        - Type of search
      - value
        - string <email>
        - Raw email address

Compromised Email API Examples

cURL

```
curl  -XPOST 'https://dev-api.mynetwatchman.com/compemail/search' \
-d
'{"search":[{"format":"sha256","value":"fe112c9f59c726d9017df1f39e62fcfa76d9ea2
c0536892a8dd5d7c52b702f5d"},{"format":"sha256", "value":
"fe112c9f59c7"},{"format":"raw","value":"test_not_user_675@example.com"},{"form
at":"sha256","value":"935b7002d54z"}]}' \
-H 'x-api-key: [PROVIDE YOUR KEY HERE]'
```

Payload


```
{
  "search": [
    {
      "format": "sha256",
      "value":
"fe112c9f59c726d9017df1f39e62fcfa76d9ea2c0536892a8dd5d7c52b702f5d"
    },
    {
      "format": "sha256",
      "value": "fe112c9f59c7"
    },
    {
      "format": "raw",
      "value": "test_not_user_675@example.com"
    },
    {
      "format": "sha256",
      "value": "935b7002d54z"
    }
  ]
}
```


Responses

∨ **200** Successful

RESPONSE SCHEMA: application/json

Results: Array of CompEmailExactMatch (object) or CompEmailPrefixMatches (object) or CompEmailError (object)

- Array [ ] Any of: CompEmailExactMatch (match), CompEmailPrefixMatches (matches), CompEmailError (error)
    - match
        - object or null
        - Matching record, or null if no records matched
            - last_seen
                - string <date-time>
                - Last time we observed the email successfully accessed
            - provider
                - string
                - Email Provider if known. If the result is "Unknown" then an MX record was not found for the domain at the time it was recorded. If the result is "Other" then the provider is not one that we have normalization rules for

                - dest_ip
                    - string
                    - The destination mail server IP address
                - dest_as
                    - string
                    - The destination mail servers Autonomous System Number (AS)
    - matches
        - Array of objects
        - Array [ ]
            - hash
                - string
                - A hash matching the prefix
            - last_seen
                - string <date-time>
                - Last time we observed the email successfully accessed
    - error
        - string
        - Description of the error that occurred
    - search
        - object (compEmailSearch)
        - The search criteria for this result (if `echo_search` was set)
        - format
            - string
            - Format of the value
        - value
            - string
            - The value to search for
    -

- 400 The client sent invalid or malformed data

- 401 The `X-API-KEY` header was not present or empty

- 403 API key was not correct, the source IP is not in the white-list, or the API key is not entitled to this API

- 405 Incorrect method was used

- 500 An internal error occurred

Response samples


```
{
  "results": [
    {
      "match": {
          "last_seen": "2022-07-05T18:25:08.000Z",
          "provider": "Other",
          "Other": "192.168.181.128",
          "dest_as": "AS64522 Private Example.com",
      }
    },
    {
      "matches": [
          {
               "hash":
"fe112c9f59c726d9017df1f39e62fcfa76d9ea2c0536892a8dd5d7c52b702f5d",
               "last_seen": "2022-07-05T18:25:08.000Z",
          }
       ]
    },
    {
      "match": null
    },
    {
      "error": "Invalid hash character 'z'"
    }
  ]
}
```


Normalize Email Addresses

POST `/compemail/normalize`

This API provides the implementation of mNW's email normalization routine.

AUTHORIZATIONS: APIKeyAuth

REQUEST BODY SCHEMA: application/json

- Array[ ]
  - Email

        - string <email>

Normalize Email Addresses API Examples

cURL

```
curl  -XPOST 'https://dev-api.mynetwatchman.com/compemail/normalize \
-d
'[{"email":"test+this@mynetwatchman.com"},{"email":"test@example.com"},{"email"
:"dummy"}]' \
-H 'x-api-key: [PROVIDE YOUR KEY HERE]'
```

Payload

```
[
    {
      "email": "test+this@mynetwatchman.com",
    },
    {
      "email": "test@example.com",
    },
    {
      "email": "dummy",
    },
]
```

Responses

\/ **200** Successful

RESPONSE SCHEMA: application/json

- Array [ ] Any of: CompEmailNormalized (normalized_email), CompEmailNormError (error)
    - normalized_email
        - object
            - verbatim
                - string
                - Email address as provided
            - provider
                - string
                - This will contain the email provider if it triggers a normalization rule. If the MX record lookup does not succeed at the time of insert it will contain "Unknown", otherwise it will be set to "Other".
            - mx

        - string
        - A single MX record for this domain. If it matches a provider rule, it will be the first record that matches the provider rule; otherwise it will be the MX record with the highest priority at the time of lookup.
    - normalized
        - string
        - Normalized email address
- error
    - string
    - Description of the error that occurred

- 400 The client sent invalid or malformed data

- 401 The `X-API-KEY` header was not present or empty

- 403 API key was not correct, the source IP is not in the white-list, or the API key is not entitled to this API

- 405 Incorrect method was used

- 500 An internal error occurred

Response samples


```
[
    {
      "normalized_email": {
          "verbatim": "test+this@mynetwatchman.com",
          "provider": "Google",
          "mx": "aspmx.l.google.com.",
          "normalized": "test@mynetwatchman.com",
      }
    },
    {
      "normalized_email": {
          "verbatim": "test@example.com",
          "provider": "Other",
          "mx": ".",
          "normalized": "test@example.com",
      }
    },
    {
      "error": "Email invalid: 'dummy'"
    }
  ]
```