Data Breach Search API

Data Breach Search NiamonX

- API - Public Breaches Search (140+ Billion Records)
- ULP (Infostealer Logs) Public Breached ULP Search | NiamonX API
- PBS v2 | Public Breached Search V2 | NiamonX API

API - Public Breaches Search (140+ Billion Records)

? NiamonX API — **Public Breaches Search (140+ Billion Records)**

Comprehensive API documentation for searching public data breaches aggregated from 4,500+ sources.

?? Important Notice

The **Public Breaches Search** API allows you to query a massive database of publicly available leaks (>140 billion records).
Use it **only** for legitimate purposes — such as verifying your own data or with explicit authorization.

> ⚠ Abuse or publication of obtained data will lead to **account suspension** and **permanent API ban**.
> Some results may contain outdated or incomplete data.

? API Endpoint

Main URL:

```
POST https://dash.niamonx.io/api/v2/breaches_search
```

Headers:

```
Content-Type: application/json
X-API-Key: YOUR_API_KEY
```

Body Example:

```
{ "query": "test@example.com" }
```

Example Request via cURL

```
curl -X POST https://dash.niamonx.io/api/v2/breaches_search \
  -H "Content-Type: application/json" \
  -H "X-API-Key: YOUR_API_KEY" \
  -d '{"query":"test@example.com"}'
```

? Query Parameters

Field	Type	Description	Example
`query`	`string`	Search term (email, phone, domain, IP, username, etc.)	`"example@mail.com"`

Supported Search Types

- Email: `john.doe@gmail.com`
- Login or Username: `nickname123`
- Domain: `domain.com`
- IP address: `1.1.1.1`
- Combo (e.g. “email password” or “name phone”)
- Full name, city, social ID

□□ **Tip:** If you get an empty response, try deleting one character from the query and repeat the request — it refreshes the search pipeline.

?? Response Structure

? When Data Is Found

HTTP 200


```
{
  "success": true,
  "data": {
    "query": "niamonx",
    "task_id": "50bab956-4a9c-4548-83ff-a0451be1b18e",
    "status": "ok",
    "detail": "file_downloaded",
    "meta": {
      "blocks_total": 3,
      "emails": ["test@niamonx.io"],
      "names": ["NiaMonx"],
      "first_seen": "2021-01-21T05:32:31+00:00",
      "last_seen": "2021-01-22T01:51:31+00:00"
    },
    "risk": {
      "score": 9,
      "level": "Low"
    },
    "blocks": [
      {
        "id": "p1",
        "title": "TestNia",
        "description": "In November 2021, the TestNia website...",
        "groups_normalized": [
          {
            "fields_map": {
              "email": "test@niamonx.io",
              "nick": "NiaMonx",
              "ip": "2800:***:9824"
            }
          }
        ]
      }
    ],
    "rate": {
      "remaining": 99,
      "reset_in_sec": 600
    },
```

```
    "cooldown_sec": 10
  }
}
```


?? When No Results Found

HTTP 200


```
{
  "success": true,
  "data": {
    "query": "not_found@niamonx.io",
    "status": "not_found",
    "detail": "no results found",
    "meta": { "blocks_total": 0 },
    "risk": { "score": 0, "level": "Low" }
  }
}
```


? When Data Is Protected

HTTP 200


```
{
  "success": true,
  "data": {
    "success": false,
    "error": "[NiamonX | DataGuard] This data has been removed and is no longer indexed by our search engine at the request of the copyright holder."
  }
}
```

? HTTP Status Codes

Code	Meaning
200	Success — request processed
400	Invalid input or malformed body
401	Invalid or missing API key
403	Tool disabled for your account
404	Unknown endpoint or invalid tool name
405	Wrong method — use `POST`
429	Cooldown or daily limit reached

? Features Summary

- □□ **140+ billion** records
- □□ **4500+** aggregated sources
- □□ **Cryptographically protected** data
- □□ **Risk Indicator** (numeric + level)
- ⚡ **Cached results** for faster response
- □□ **Automatic data removal** for sensitive categories (bank/medical)

? Code Examples

1. Python (requests)

```
import requests

url = "https://dash.niamonx.io/api/v2/breaches_search"
headers = {
    "Content-Type": "application/json",
    "X-API-Key": "YOUR_API_KEY"
}
```

```
payload = { "query": "example@mail.com" }

response = requests.post(url, json=payload, headers=headers)
print(response.status_code)
print(response.json())
```

2. JavaScript (Node.js / Axios)

```
import axios from "axios";

const API_KEY = "YOUR_API_KEY";
const url = "https://dash.niamonx.io/api/v2/breaches_search";

async function searchLeak(query) {
  try {
    const res = await axios.post(
      url,
      { query },
      {
        headers: {
          "Content-Type": "application/json",
          "X-API-Key": API_KEY
        }
      }
    );
    console.log(res.data);
  } catch (err) {
    console.error("Error:", err.response?.data || err.message);
  }
}

searchLeak("example@mail.com");
```

3. PHP (cURL)

```php
<?php
$api_key = "YOUR_API_KEY";
$url = "https://dash.niamonx.io/api/v2/breaches_search";

$data = ["query" => "example@mail.com"];
$options = [
  CURLOPT_URL => $url,
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_POST => true,
  CURLOPT_HTTPHEADER => [
    "Content-Type: application/json",
    "X-API-Key: $api_key"
  ],
  CURLOPT_POSTFIELDS => json_encode($data)
];

$ch = curl_init();
curl_setopt_array($ch, $options);
$response = curl_exec($ch);
curl_close($ch);

echo $response;
?>
```

---

4. Go (net/http)

```go
package main

import (
"bytes"
"fmt"
"io"
"net/http"
)
```

```
func main() {
	apiKey := "YOUR_API_KEY"
	jsonBody := []byte(`{"query":"example@mail.com"}`)
	req, _ := http.NewRequest("POST", "https://dash.niamonx.io/api/v2/breaches_search",
bytes.NewBuffer(jsonBody))
	req.Header.Set("Content-Type", "application/json")
	req.Header.Set("X-API-Key", apiKey)

	client := &http.Client{}
	resp, err := client.Do(req)
	if err != nil {
		panic(err)
	}
	defer resp.Body.Close()

	body, _ := io.ReadAll(resp.Body)
	fmt.Println("Status:", resp.Status)
	fmt.Println(string(body))
}
```

? Best Practices

- Wait **3 seconds** between requests to respect cooldown (anti-spam).
- Combine parameters intelligently:
  - `"email password"`
  - `"name phone"`
- Do not republish or share results publicly.
- If you detect compromised credentials — **change passwords immediately**.
- Sensitive datasets (bank cards, healthcare) are **automatically excluded**.

? Summary

Feature	Description
Endpoint	`https://dash.niamonx.io/api/v2/breaches_search`
Method	POST

Feature	Description
Authentication	`X-API-Key`
Cooldown	3 seconds
Rate Limit Info	Returned in `rate` object
Data Types Supported	Email, phone, IP, domain, username
Data Protection	SHA-256 & DataGuard Compliance

 Now you can safely integrate the NiamonX Public Breaches API into your OSINT tools, cybersecurity dashboards, or monitoring systems — with full ethical and legal compliance.

ULP (Infostealer Logs) Public Breached ULP Search | NiamonX API

? NiamonX API — **ULP (Infostealer Logs) Public Breached Search**

Comprehensive API documentation for searching **ULP datasets (URL · LOGIN · PASSWORD)** extracted from public infostealer logs and credential leaks.

? What Is ULP?

ULP stands for **URL · LOGIN · PASSWORD** — a triple that represents evidence of compromised credentials captured in stealer logs or public leaks.

Each ULP record links:

- **URL:** the website or endpoint where the credential was used (e.g., `example.com/login`)
- **LOGIN:** the username or email associated with the site
- **PASSWORD:** the stolen or leaked password (masked by default for privacy)

This API enables you to search across massive datasets for specific entries by **email**, **username**, **domain**, **URL**, or **password**.

?? Usage & Ethics

By using this endpoint, you confirm that:

- You **own** the data or have explicit permission to process it.
- You will **not share** results publicly or misuse obtained data.
- You will **act responsibly** — change any compromised credentials and enable MFA.

All queries are **end-to-end encrypted**, and NiamonX **never logs or shares** search data.

? Endpoint

```
POST https://dash.niamonx.io/api/v2/ulp_search
```

? Request Structure

Headers

```
Content-Type: application/json
X-API-Key: YOUR_API_KEY
```

Body Example

```
{ "action": "search", "value": "test@example.com", "type": "auto", "exact": true, "limit": 200
}
```

?? Request Parameters

Field	Type	Description	Example
action	`string`	Must be `"search"`	`"search"`
value	`string`	Your search query (email, domain, password, etc.)	`"user@mail.com"`
type	`string`	Search type (see below)	`"email"`

Field	Type	Description	Example
exact	`boolean`	Whether to match exactly (recommended for emails)	`true`
limit	`integer`	Max number of records (1–1000)	`200`

Available `type` values:

- `auto` *(default)*
- `email`
- `username`
- `domain`
- `url`
- `password`

? Example cURL Request

```
curl -X POST https://dash.niamonx.io/api/v2/ulp_search \
  -H "Content-Type: application/json" \
  -H "X-API-Key: YOUR_API_KEY" \
  -d '{"action":"search","value":"test@example.com","type":"auto","exact":true,"limit":200}'
```

? Successful Response

HTTP 200


```
{
  "success": true,
  "data": {
    "query": {
      "value": "test@niamonx.io",
      "type": "email",
```

```
      "exact": true,
      "limit": 200
    },
    "stats": {
      "total": 1,
      "unique_hosts": 1,
      "with_password": 1
    },
    "records": [
      {
        "id": "ac22b9424f8aab6011fb526c9798e7c3898652d4c7a6eb8c0253212d94a9fec4",
        "url": "niamonx.io/login/index",
        "host": "niamonx.io",
        "login": "test@niamonx.io",
        "pass": "NiaMon750H",
        "score": 8.840368
      }
    ],
    "status": "ok",
    "cached": false,
    "fetched_at": "2025-11-09T21:50:31+00:00",
    "api_timing_ms": 75
  }
}
```

?? When Nothing Is Found

HTTP 200


```
{
  "success": true,
  "data": {
    "query": {
      "value": "not_found",
      "type": "username",
```

```
      "exact": true,
      "limit": 200
    },
    "stats": {
      "total": 0,
      "unique_hosts": 0,
      "with_password": 0
    },
    "records": [],
    "status": "ok",
    "cached": false,
    "fetched_at": "2025-11-09T21:51:56+00:00",
    "api_timing_ms": 63
  }
}
```

? Protected Data Response

HTTP 200


```
{
  "success": true,
  "data": {
    "success": false,
    "error": "[NiamonX | DataGuard] This data has been removed and is no longer indexed by our search engine at the request of the copyright holder."
  }
}
```

? HTTP Status Codes

Code	Description

200	Success — request processed
400	Invalid input or malformed request
401	Invalid or missing API key
403	Tool disabled for your account
404	Unknown endpoint or invalid tool
405	Wrong HTTP method (use `POST`)
429	Cooldown or daily limit exceeded (ToolService message)

? Tips & Notes

- □ **Limit:** up to 1000 results per request
- □□ **Cooldown:** few seconds between requests to prevent spam
- □□ **Domain search:** finds subdomains automatically
- □□ **Duplicate removal** and periodic reindexing ensure fresh results
- □□ If results seem incomplete, retry in several minutes for updated data

---

? Code Examples

1. Python (requests)

```
import requests
import json

url = "https://dash.niamonx.io/api/v2/ulp_search"
headers = {
    "Content-Type": "application/json",
    "X-API-Key": "YOUR_API_KEY"
}
payload = {
    "action": "search",
    "value": "test@example.com",
    "type": "auto",
    "exact": True,
```

```
    "limit": 200
}

response = requests.post(url, headers=headers, json=payload)
print(response.status_code)
print(json.dumps(response.json(), indent=2))
```

---

2. JavaScript (Node.js / Axios)

```
import axios from "axios";

const API_KEY = "YOUR_API_KEY";
const url = "https://dash.niamonx.io/api/v2/ulp_search";

async function searchULP(query) {
  try {
    const res = await axios.post(
      url,
      {
        action: "search",
        value: query,
        type: "auto",
        exact: true,
        limit: 200
      },
      {
        headers: {
          "Content-Type": "application/json",
          "X-API-Key": API_KEY
        }
      }
    );
    console.log(JSON.stringify(res.data, null, 2));
  } catch (err) {
    console.error("Error:", err.response?.data || err.message);
  }
```

```
}

searchULP("test@example.com");
```

---

3. PHP (cURL)

```php
<?php
$apiKey = "YOUR_API_KEY";
$url = "https://dash.niamonx.io/api/v2/ulp_search";

$data = [
  "action" => "search",
  "value" => "test@example.com",
  "type" => "auto",
  "exact" => true,
  "limit" => 200
];

$options = [
  CURLOPT_URL => $url,
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_POST => true,
  CURLOPT_HTTPHEADER => [
    "Content-Type: application/json",
    "X-API-Key: $apiKey"
  ],
  CURLOPT_POSTFIELDS => json_encode($data)
];

$ch = curl_init();
curl_setopt_array($ch, $options);
$response = curl_exec($ch);
curl_close($ch);

echo $response;
?>
```

4. Go (net/http)

```
package main

import (
	"bytes"
	"fmt"
	"io"
	"net/http"
)

func main() {
	apiKey := "YOUR_API_KEY"
	body := []byte(`{"action":"search","value":"test@example.com","type":"auto","exact":true,"limit":200}`)

	req, _ := http.NewRequest("POST", "https://dash.niamonx.io/api/v2/ulp_search", bytes.NewBuffer(body))
	req.Header.Set("Content-Type", "application/json")
	req.Header.Set("X-API-Key", apiKey)

	client := &http.Client{}
	resp, err := client.Do(req)
	if err != nil {
		panic(err)
	}
	defer resp.Body.Close()

	result, _ := io.ReadAll(resp.Body)
	fmt.Println("Status:", resp.Status)
	fmt.Println(string(result))
}
```

? Summary

Feature	Description
Endpoint	`https://dash.niamonx.io/api/v2/ulp_search`
Method	`POST`
Auth Header	`X-API-Key`
Limit	up to 1000 records
Cooldown	Short delay between requests
Sensitive Data	Passwords masked, removed on request
Security	Encrypted E2E, DataGuard compliant
Data Source	Public infostealer logs and breach repositories

 With the ULP API, you can ethically and securely analyze exposure of credentials from public infostealer datasets — empowering your investigations, threat intelligence, and personal data protection workflows.

PBS v2 | Public Breached Search V2 | NiamonX API

? Public Breached Search V2 — NiamonX API Guide

The **Public Breached Search V2** endpoint allows you to search for publicly available breach records in the NiamonX secure data analysis system.
It operates through a **private, encrypted channel** and uses a **minimal indexing model** to protect sensitive data.

Supported identifiers include:

- **Email**
- **Username**
- **Phone**
- **Hash**

All requests must be made securely and responsibly.
Never share or publish personal results — doing so may violate data protection laws and lead to account suspension.

? API Endpoint

```
POST https://dash.niamonx.io/api/v2/breaches_s_v2
```

Headers:

```
Content-Type: application/json
X-API-Key: YOUR_API_KEY
```

Body:

```
{ "value": "test@example.com", "type": "auto" }
```

Available types:
`auto`, `email`, `username`, `phone`, `hash`

? Description

This API searches **closed and anonymized datasets** for breached credential evidence.
Decryption happens at the client level using your master key, ensuring full end-to-end security.

Additional features:

- Passwords are hidden by default (toggleable).
- Supports export to CSV/JSON.
- Local query history.
- Internal quotas are invisible to users.
- Misuse or unauthorized data searches may result in a ban.

? Example Successful Response


```
{
  "success": true,
  "data": {
    "query": {
      "value": "test@niamonx.io",
      "type": "auto"
    },
    "stats": {
      "found": 2,
      "returned": 2,
      "with_passwords": 2,
      "unique_sources": 1,
      "earliest_month": null,
      "latest_month": null
    },
```

```
"records": [
  {
    "source": {
      "name": "Stealer Logs",
      "breach_date": null,
      "unverified": 0,
      "passwordless": 0,
      "compilation": 1
    },
    "account": "test@niamonx.io",
    "email": "test@niamonx.io",
    "username": null,
    "phone": null,
    "hash": null,
    "password": "z8NiAmOn50H",
    "fields": [
      "password",
      "origin",
      "email"
    ]
  },
  {
    "source": {
      "name": "Stealer Logs",
      "breach_date": null,
      "unverified": 0,
      "passwordless": 0,
      "compilation": 1
    },
    "account": "test@niamonx.io",
    "email": "test@niamonx.io",
    "username": null,
    "phone": null,
    "hash": null,
    "password": "ViptraNiA!",
    "fields": [
      "password",
      "origin",
      "email"
    ]
```

```
      }
    ],
    "niamonx_success": true,
    "status": "ok",
    "fetched_at": "2025-11-09T22:04:00+00:00",
    "api_timing_ms": 146
  }
}
```

?? Possible Outcomes

No Matches Found


```
{
  "success": true,
  "data": {
    "success": false,
    "status": "error",
    "error": "HTTP 400",
    "query": {
      "value": "not_found",
      "type": "auto"
    }
  }
}
```


DataGuard Protection


```
{
  "success": true,
  "data": {
    "success": false,
    "error": "[NiamonX | DataGuard] This data has been removed and is no longer indexed by our
```

```
search engine at the request of the copyright holder."
  }
}
```

? HTTP Status Codes

Code	Meaning
200	✅ Successful response (check data object)
400	❌ Validation error in input data
401	🔑 Invalid or missing API key
403	🚫 Tool disabled or unavailable
404	❓ Unknown endpoint
405	⚙️ Method not allowed (use `POST`)
429	⏳ Cooldown / daily limit reached

? Example Implementations

1. cURL

```
curl -X POST https://dash.niamonx.io/api/v2/breaches_s_v2 \
  -H "Content-Type: application/json" \
  -H "X-API-Key: YOUR_API_KEY" \
  -d '{"value":"test@example.com","type":"auto"}'
```

---

2. Python (using `requests`)

```
import requests

url = "https://dash.niamonx.io/api/v2/breaches_s_v2"
headers = {
    "Content-Type": "application/json",
    "X-API-Key": "YOUR_API_KEY"
}
data = {
    "value": "test@example.com",
    "type": "auto"
}

response = requests.post(url, json=data, headers=headers)
print(response.json())
```

---

3. JavaScript (Node.js, using `axios`)

```
import axios from "axios";

const url = "https://dash.niamonx.io/api/v2/breaches_s_v2";
const headers = {
  "Content-Type": "application/json",
  "X-API-Key": "YOUR_API_KEY"
};

const data = {
  value: "test@example.com",
  type: "auto"
};

axios.post(url, data, { headers })
  .then(res => console.log(res.data))
  .catch(err => console.error(err.response?.data || err.message));
```

---

4. PHP

```php
<?php
$url = "https://dash.niamonx.io/api/v2/breaches_s_v2";
$headers = [
    "Content-Type: application/json",
    "X-API-Key: YOUR_API_KEY"
];
$body = json_encode([
    "value" => "test@example.com",
    "type" => "auto"
]);

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $body);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

$response = curl_exec($ch);
curl_close($ch);

echo $response;
?>
```

---

5. Go

```go
package main

import (
"bytes"
"encoding/json"
"fmt"
"net/http"
"io/ioutil"
)
```

```
func main() {
	url := "https://dash.niamonx.io/api/v2/breaches_s_v2"
	body := map[string]string{
		"value": "test@example.com",
		"type":  "auto",
	}
	jsonData, _ := json.Marshal(body)

	req, _ := http.NewRequest("POST", url, bytes.NewBuffer(jsonData))
	req.Header.Set("Content-Type", "application/json")
	req.Header.Set("X-API-Key", "YOUR_API_KEY")

	client := &http.Client{}
	resp, _ := client.Do(req)
	defer resp.Body.Close()

	respBody, _ := ioutil.ReadAll(resp.Body)
	fmt.Println(string(respBody))
}
```

?? Security Recommendations

- Keep your API key private; never hard-code it in shared repositories.
- Use HTTPS exclusively — no plain HTTP connections are allowed.
- If your key is compromised, revoke it immediately.
- Store credentials in environment variables or encrypted vaults.
- Do not reshare or republish search results publicly.

 You’re all set!
With your **NiamonX API Key** and this **Public Breached Search V2** guide, you can perform encrypted and privacy-compliant breach lookups safely and efficiently.