DNSDB™ API

Technical Datasheet

DNSDB

The DNSDB™ is built from Farsight Security's Passive DNS data. In addition to the most recent information, DNSDB contains historical data going back to 2010. Farsight stores and indexes two types of data:

- Observed DNS data, as recorded by Farsight's network of globally distributed sensors, and
- DNS records gleaned from tld zone files.

DNSDB makes it easy to find related domain names and IP addresses—assuming you have an initial domain name or IP address as a starting point. DNSDB can answer questions, such as:

- Where did this domain name point to in the past?
- What domain names are hosted on a given IP address?
- What domain names use a given nameserver?
- What fully qualified domain names exist below a delegation point?

What is Passive DNS?

"Passive DNS" or "Passive DNS replication" is a technique invented by Florian Weimer in 2004 to opportunistically reconstruct a partial view of the data available in the global Domain Name System into a central database where it can be indexed and queried. During the initial collection stage, packets between DNS resolvers and authoritative DNS servers are collected together at a central processing point. There are many steps required to proces passive DNS data into a cohesive DNS database that can be queried. For more information, access *What is Passive DNS?* - https://www.farsightsecurity.com/technical/passive-dns/passive-dns-faq/#q11

Limitations of Regular DNS

"Regular" DNS is a one-way view. It only shows what's currently configured for a particular domain or IP address. It doesn't show many key relationships, nor does it provide a historical view. DNSDB "synthesizes" many latent or implicit DNS relationships, and thereby enables a broader range of queries. For example, DNSDB makes it possible to find virtually all the domains using the same nameserver. Or, given an IP address, DNSDB can tell a user all the hostnames that have historically been seen using that IP address. DNSDB can do these things and much more.

How Can DNSDB Be Used?

There are many ways that security teams can use DNSDB. Some ways include:

- **Identify shared infrastructure:** Users may be confronting unwanted traffic from a given source. Should they block it? If they do so, and the IP is used for shared hosting, there may be undesirable collateral damage. DNSDB will let the analyst tell what's been seen on that IP over time.
- **Validate DNS configuration:** IT teams may need to identify changes when they happen for a particular host or domain. DNSDB can be used to track these movements—even if the adversary is using a fast flux domain.
- **Identify related infrastructure:** DNSDB is all about making connections. DNSDB lets a security analyst find related domains using the same name server infrastructure used by a "known bad" domain, thereby avoiding incomplete take downs.
- **Conduct research and analysis:** DNSDB also helps you do a post-hoc analysis of some operational issues. Why did a given service break? Was there a DNS server administrator typo?

Why Does DNSDB Work?

The internet relies heavily on the DNS, and criminals are not exempt. DNSDB exploits the fact that cyber criminals share and reuse resources. This can include sharing name servers or sharing IP addresses. Other times cyber criminals will obtain a consecutive range of IP addresses, so finding one will often let you locate additional bad addresses in the same network neighborhood or netblock. In this regard, criminal activity can leave footprints in DNS. Using DNSDB, investigators can follow those trails even after the criminals have covered their tracks in regular DNS.

DNSDB Access Methods

Farsight's DNSDB can be accessed via:

- DNSDB API
- Third party software integration (using the DNSDB API)
- DNSDB Export (DNSDB with a copy of the DNSDB data located "on premises")

This document focuses on DNSDB API. This datasheet does not cover DNSDB Export. For more information on these products, *contact Farsight* - https://www.farsightsecurity.com/about-farsight-security/contacts/

DNSDB API

Farsight's DNSDB API service can be accessed many different ways. Farsight supports key portability allowing customers to use their purchased key with many different tools. Access methods include:

Delivery Method	Target Audiences
API for developers	Targets security application developers who can integrate DNSDB API into an existing application, or write their own front-end interface to DNSDB's API
Company-provided demo command line clients (dnsdbq, dnsdb_query.py, etc.)	Security, incident response, SOC, and research teams can easily leverage the power of DNSDB API without having to be a programmer
Third party software integrations (such as Splunk App and Maltego Transforms)	Provides a convenient way for analysts to easily/automatically enhance data managed by their favorite existing tools

DNSDB API for Developers

The following section provides details that a developer would need to integrate DNSDB API into their programs.

Service Endpoint

DNSDB API is provided over HTTPS at https://api.dnsdb.info/

Access Control / Authentication

Access is limited to subscribers. To request an API key, please complete the Farsight Security service application form at *Get Started* - https://www.farsightsecurity.com/get-started/

Subscribers authenticate with a Farsight-provided DNSDB API authentication key. This key is provided by the developer in an X-API-Key header. For example, if your API key is d411234567890123456789012345678 9, then the following header should be added your request:

```
X-API-Key: d41123456789012345678901234567 89
```

If the X-API-Key header is not present, or the provided API key is not valid, a 403 Forbidden response will be returned to your client.

Service Limits

1. By default, DNSDB API users are limited to 10 concurrent connections.
2. Each API key has an associated quota. That quota limits the number of DNSDB API requests that can be made. There are three types of quotas: time-based, block-based, and unlimited.
    - Time-based quotas are usually applied on a daily basis and resets daily at 00:00 (midnight) in the UTC time zone. Time-based quotas can also be applied for arbitrary time-quantum, but this is unusual. Customers requiring more flexibility may opt for a block-based quotas, which have a specific number of queries with an expiration time (the expiration time is usually much longer than a day).
    - Block quota customers must purchase additional quota when their quota reaches zero or expires. There are no daily query restrictions for block quota, allowing investigators to process a large number of queries in a single day. A representative block quota would range from 100,000 up to 5 million queries. With either a time-based quota or a block-based quota, when the quota reaches zero, users receive a response letting them know their quota limit is exceeded. Unused time-based quotas do not rollover. Block quota customers may be able to rollover their unused queries at the end of their subscription period if they renew or purchase a new subscription **prior to expiration**, otherwise, unused block quota queries that expire are lost forever.

The /lookup/rate_limit endpoint returns a JSON map containing a top-level map named rate that contains a set of numeric keys depending on the type of subscription. The **Reset**, **Limit**, and **Remaining** fields are always provided. The **Expires** field is only shown when the API key is for a block quota.

Key	Description
Reset	UNIX epoch timestamp with second granularity indicating the next point in time when the quota limit will be reset — Usually 00:00 (midnight) UTC
Expires	UNIX epoch timestamp indicating the expiration date and time of the block quota
Limit	Maximum number of API lookups that may be performed during the quota period
Remaining	Number of **remaining** API lookups that may be performed during the quota period

Service Limit Response Codes

The following is an example of a /lookup/rate_limit response that indicates that the API key's quota limit will be reset at midnight UTC, and that 999 lookups are remaining out of a total quota of 1000 lookups:

```
{
   "rate": {
      "reset": 1433980800,
      "limit": 1000,
      "remaining": 999
   }
}
```

The following is an example of a /lookup/rate_limit response for an API key associated with a block quota assigned 150,000 queries with 149,851 remaining. Since there is no reset for a block quota, the value is "n/a". Some clients are able to translate the epoch time to a more readable date time format. The second version is the output of dnsdbq -I to check the quota which defaults to text format.

```
{
 "rate": {
   "reset": "n/a",
   "expires": 1569888000,
   "limit": 150000,
   "remaining": 149851
 }
}
```

Text format with easy to read date value from dnsdbq -I

```
quota:
      reset: n/a
      expires: 2019-10-01 00:00:00
      limit: 150000
      remaining: 149851
```

For API keys that have no quota limit configured ("an unlimited API key"), the response will set the three rate fields to the value -1, i.e.:

```
{
   "rate": {
      "reset": -1,
      "limit": -1,
      "remaining": -1
   }
}
```

Querying the /lookup/rate_limit endpoint does not count against the quota limit!

Responses from the /lookup/rrset and /lookup/rdata endpoints contain the same information in the HTTP response headers that can be obtained from the /lookup/rate_limit endpoint. These values are embedded as the X-RateLimit-Limit, X-RateLimit-Remaining, and X-RateLimit-Reset headers. For a block quota, an additional X-Ratelimit-Expires header is included. For unlimited API keys, the fields will be encoded with the strings “unlimited” (for X-RateLimit-Limit) and “n/a” (for X-RateLimit-Remaining and X-RateLimit-Reset). Otherwise, for API keys with a quota limit defined, the values for these headers will be encoded numerically as described above for the /lookup/rate_limit endpoint. Responses to lookups will contain response headers that look like this:

Example headers for a per day quota:

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 999
X-RateLimit-Reset: 1433980800
```

Example headers for a block quota:

```
X-RateLimit-Limit: 150000
X-RateLimit-Remaining: 149851
X-RateLimit-Reset: n/a
X-RateLimit-Expires: 1569888000
```

Or, in the case of an unlimited API key:

```
X-RateLimit-Limit: unlimited
X-RateLimit-Remaining: n/a
X-RateLimit-Reset: n/a
```

Query Structure

This section explains the formation of DNSDB API queries element by element. DNSDB API query URLs all begin with https://api.dnsdb.info/

Lookup Methods

All DNSDB lookup requests are rooted in URL paths under the /lookup hierarchy. There are two separate lookup methods, “rrset” and “rdata”, which are accessed via the URL paths /lookup/rrset and /lookup/rdata respectively.

- RRset queries search DNSDB for matches in the “left-hand side” of DNSDB DNS records
- Rdata queries search DNSDB for matches in the “right-hand side” of DNSDB DNS records

For example: www.farsightsecurity.com IN A 66.160.140.81

Left-hand side	Right-hand side
www.farsightsecurity.com	66.160.140.81

Lookup Method	Description
RRset queries	Only searches the left-hand side for matches. Returns records in DNSDB associated with a specific host or a wildcarded domain. Use when you have a domain name and want to search the left-hand side.
Rdata queries	Only searches the right-hand side for matches. Returns a list of domain names found in DNSDB which rely on the specified host for name service **or** returns DNSDB records matching a specific IP address (or CIDR netblock) or range of IP addresses. Use when you have a domain name, an IP address (or CIDR netblock), or a range of IP addresses and want to search the right-hand side.

For more information on RRset and Rdata queries, access *RRset and Data Demystified* - https://www.farsightsecurity.com/2015/03/11/stsauver-rrset-rdata/

rrset Lookups

The **rrset** lookup queries within DNSDB's RRset index supports "forward" lookups based on the owner name ("domain name") of an RRset.

An owner name with a leading asterisk and label separator, (i.e., "*.") performs a wildcard search for any RRsets whose owner names end with the given domain name. An owner name with a trailing label separator and asterisk (i.e., ".*") performs a wildcard search for any RRsets whose owner names start with the given label(s). Note that the latter type of query is somewhat more computationally laborious for DNSDB to run due to how DNSDB's data is stored.

Partial label queries are not supported; a search has to be done within a domain (e.g. *.example.com) or for a complete domain component (e.g. example.*). Sub-string searches such as *ample.com or ex*le.com do not work. More information on this is available in *RRset and Data Demystified* - https://www.farsightsecurity.com/2015/03/11/stsauver-rrset-rdata/

rrset lookups can optionally be filtered based on RRtype and/or bailiwick.

rrset lookups follow the URL path scheme:

```
/lookup/rrset/name/$owner_name/$rrtype/$bailiwick
```

As with rrsets, the first portion of the URL (/lookup/rrsets/name) is static and should be written as is. Placeholders have been included for owner name, rrtype, and bailiwick. Possible formats would be:

Possible formats would be:

```
/lookup/rrset/name/$owner_name
/lookup/rrset/name/$owner_name/$rrtype
/lookup/rrset/name/$owner_name/$rrtype/$bailiwick
```

For example:

```
/lookup/rrset/name/example.com
/lookup/rrset/name/example.com/NS
/lookup/rrset/name/example.com/NS/com
```

Note that some elements are required (see table below).

Element	Required?	Notes
owner_name	Yes	*owner_name* is the DNS name specified in DNS presentation format.
rrtype	No	*rrtype* is specified as a DNS RRtype mnemonic. rrtype **ANY** may be specified for rrtype in order to perform bailiwick filtering without also filtering on a particular RRtype. The ANY record type is modified somewhat from its usual meaning. A DNSDB lookup for RRtype ANY will match any RRtype except the DNSSEC-related RRtypes: DS, RRSIG, NSEC, DNSKEY, NSEC3, NSEC3PARAM, and DLV. A new pseudo-mnemonic ANY-DNSSEC has been introduced that will return only those records matching the aforementioned RRtypes.
bailiwick	No	For more information on *bailiwicks*, visit *What is a Bailiwick?* - https://www.farsightsecurity.com/2017/03/21/stsauver-what-is-a-bailiwick/

rdata Lookups

The **rdata** lookup queries DNSDB's Rdata field, which supports "inverse" lookups based on Rdata record values. In contrast to the rrset lookup method, rdata lookups return only individual resource records and not full resource record sets, and rdata results lack bailiwick metadata. If the full RRset and bailiwick metadata are required, additional RRset queries will need to be done on the results obtained from the rdata query.

rdata lookups follow the URL path scheme:

```
/lookup/rdata/$type/$value/$rrtype
```

As with rrsets, the first portion of the URL (/lookup/rdata) is static and should be written as is. Placeholders have been included for type, value, and rrtype. Possible formats would be:

```
/lookup/rdata/$type
/lookup/rdata/$type/$value
/lookup/rdata/$type/$value/$rrtype
```

Note that some elements are required (see table below).

Element	Required?	Notes
type	Yes	Specifies how *value* is interpreted. Type may be one of the following: name - The *value* is a DNS domain name in presentation format, searching for a sub-domain ("*.example.com") or top-level domain ("www.example.*") wildcard domain name. Note that sub-domain wildcard queries are somewhat more resource intensive than top-level domain wildcard queries. ip - The *value* is one of an IPv4 address, an IPv6 address, an IPv4 network with prefix length, or an IPv6 network with prefix length. If a network lookup is being performed, the delimiter between network address and prefix length is a single comma (",") character rather than the usual slash ("/") character to avoid clashing with the HTTP URL path name separator. raw - The *value* is an even number of hexadecimal digits specifying a raw octet string. For more information on Raw Hex, visit *Raw Hex Rdata Queries* - https://www.farsightsecurity.com/2016/11/25/stsauver-dnsdb-rawhex/
value	Yes	Represents the data you want to look up—domain name, IP address, or IP range. For more information on Raw Hex, visit *Raw Hex Rdata Queries* - https://www.farsightsecurity.com/2016/11/25/stsauver-dnsdb-rawhex/
rrtype	No	Optionally filters the results by RRtype in the same manner as the rrset lookup.

Output Formats

Results can be provided in presentation format or JSON lines format. The format to be used for your output is chosen by supplying an appropriate Accept header as part of your request:

```
Accept: application/json

or

Accept: text/plain
```

If no Accept header is provided, presentation format will be used by default.

Presentation Format

The **rrset** text result format is reminiscent of the DNS master file format. Each result is an RRset separated by two newline characters and is annotated with bailiwick and timestamp information, prefixed with leading ;; characters.

At the end of the result document is a footer prefixed with leading ;; characters. Currently the footer only notes how many results were found and how long it took to query the database.

Note that multiple RRsets with the same owner name and RRtype can appear in the result document.

The **rdata** text result format is similar, except that full RRsets are not returned (only resource records), no bailiwick metadata is provided, and individual results are separated by a single newline rather than two.

Sample Call: Text Result

```
curl -H 'Accept: application' -H 'X-API-Key: <elid>' "https://api.dnsdb.info/lookup/rrset/name/www.farsightsecurity.com/A"

;;  bailiwick: farsightsecurity.com.
;;      count: 5059
;; first seen: 2013-09-25 20:02:10 -0000
;;  last seen: 2015-04-01 09:51:39 -0000
www.farsightsecurity.com. IN A 66.160.140.81

;;  bailiwick: farsightsecurity.com.
;;      count: 44151
;; first seen: 2015-04-01 13:07:24 -0000
;;  last seen: 2018-03-12 19:57:23 -0000
www.farsightsecurity.com. IN A 104.244.13.104

;;  bailiwick: farsightsecurity.com.
;;      count: 164
;; first seen: 2013-07-01 17:37:26 -0000
;;  last seen: 2013-09-24 17:14:08 -0000
www.farsightsecurity.com. IN A 149.20.4.207

;;; Returned 3 RRsets in 0.02 seconds.
;;; DNSDB
```

JSON Format

The rrset JSON result format is a document containing one JSON-encoded result object per line. Each result object is an associative array with the following keys:

Key	Description
rrname	The owner name of the RRset in DNS presentation format.
rrtype	The resource record type of the RRset, either using the standard DNS type mnemonic, or an RFC 3597 generic type (i.e., the string TYPE immediately followed by the decimal RRtype number).
rdata	An array of one or more Rdata values. The Rdata values are converted to the standard presentation format based on the rrtype value. If the encoder lacks a type-specific presentation format for the RRset's rrtype, then the RFC 3597 generic Rdata encoding will be used.
bailiwick	For more information on bailiwicks, visit *What is a Bailiwick?* - https://www.farsightsecurity.com/2017/03/21/stsauver-what-is-a-bailiwick/
count	The number of times the RRset was observed via passive DNS replication.
time_first, time_last	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via passive DNS replication. Will not be present if the RRset was only observed via zone file import.
zone_time_first, zone_time_last	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via zone file import. Will not be present if the RRset was only observed via passive DNS replication.

The metadata at the bottom of the results includes:

Metadata	Description
Count	The number of times the RRset was observed via passive DNS replication.
Bailiwick	The "bailiwick" of an RRset in DNSDB observed via passive DNS replication is the closest enclosing zone delegated to a nameserver which served the RRset; The "bailiwick" of an RRset in DNSDB observed in a zone file is simply the name of the zone containing the RRset.
First Seen	UTC timestamp with seconds granularity indicating the *first time* an RRset was seen in the given bailiwick via passive DNS replication.
Last Seen	UTC timestamp with seconds granularity indicating the *last time* an RRset was seen in the given bailiwick via passive DNS replication.
First Seen in Zone File	UTC timestamp with seconds granularity indicating the *first time* an RRset was seen in the given bailiwick via zone file import.
Last Seen in Zone File	UTC timestamp with seconds granularity indicating the *last time* an RRset was seen in the given bailiwick via zone file import.

An rrset search result may be missing either the pair of (first seen, last seen) timestamps from passive DNS replication **or** from zone file import. However, there will always be one pair of timestamps present with an rrset search. This may change if a fundamentally new data source is introduced in the future, but there will always be at least one timestamp pair associated with an RRset.

rdata Results

Key	Description
rrname	The owner name of the resource record in DNS presentation format.
rrtype	The resource record type of the resource record, either using the standard DNS type mnemonic, or an RFC 3597 generic type, i.e. the string TYPE immediately followed by the decimal RRtype number.
rdata	The record data value. The Rdata value is converted to the standard presentation format based on the rrtype value. If the encoder lacks a type-specific presentation format for the resource record's type, then the RFC 3597 generic Rdata encoding will be used.
count	The number of times the resource record was observed via passive DNS replication.
time_first, time_last	UNIX epoch timestamps with second granularity indicating the first and last times the resource record was observed via passive DNS replication.
zone_time_first, zone_time_last	UNIX epoch timestamps with second granularity indicating the first and last times the resource record was observed via zone file import.

Sample Call: JSON Result

```
curl -H 'Accept: application/json' -H 'X-API-Key: <elid>' "https://api.dnsdb.info/lookup/rrset/name/www.farsightsecurity.com/A"

{"count":5059,"time_first":1380139330,"time_last":1427881899,"rrname":"www.farsightsecurity.com.","rrtype":"A","bailiwick":
"farsightsecurity.com.","rdata":["66.160.140.81"]}
{"count":44151,"time_first":1427893644,"time_last":1520884643,"rrname":"www.farsightsecurity.com.","rrtype":"A","bailiwick"
:"farsightsecurity.com.","rdata":["104.244.13.104"]}
{"count":164,"time_first":1372700246,"time_last":1380042848,"rrname":"www.farsightsecurity.com.","rrtype":"A","bailiwick":"f
arsightsecurity.com.","rdata":["149.20.4.207"]}
```

Max Results Returned

There is a built-in limit to the number of results that are returned via these lookup methods. The default limit is set at ten thousand (10,000). This limit can be raised or lowered by setting the "limit" query parameter. E.g., appending "?limit=20000" to the URL path will set the response limit to 20,000 results. The maximum number of results you can request is 1,000,000.

Time Fencing

Results can be filtered by time using the "time_first_before," "time_first_after," "time_last_before," and "time_last_after" query parameters. These parameters expect a integer (Unix Epoch time) with seconds granularity or a relative time in seconds (preceded by `-`).

Return Codes

When a query is submitted, a return code will be generated. The following is a list DNSDB API return codes.

Return Code	Description
200 OK	Successful request is returned
400 Bad Request	URL is formatted incorrectly
401 Unauthorized	The API key is not authorized (usually indicates the block quota is expired)
403 Forbidden	X-API-Key header is not present or the provided API key is not valid
404 Not Found	No records found for the given lookup
429 Too Many Requests	API key quota limit is exceeded
500 Internal Server Error	Error processing the request
503 Service Unavailable	Too many concurrent connections
504 Gateway Time-out	Backend database was unable to return all results to the front end API in time; often a sign of an incorrectly formed query asking for hundreds of millions of results, etc.

System Requirements

DNSDB API can be accessed from any Internet connected host. DNSDB API requires the ability to create a TLS-secured “https” RESTful connection. These connections can be established from software/systems of the user’s choice.

Dependencies

As an API, there are very few absolute dependencies other than network access. DNSDB API is delivered via a RESTful HTTPS API and requires HTTPS access. Network connectivity issues, outbound firewalls, or air-gapped networks could prevent or limit access to the API server. **Note:** If an un-encrypted transport is required, DNSDB-Export is the only option.

Most of the dependencies for DNSDB are based on the client software integrating DNSDB. This can be anything from Splunk or Maltego integration tools, to a custom-built integration. Refer to specific dependencies and requirements for that software.

When working with the command line reference client, you must have a unix shell. Each of the API command line clients has its own requirements for being installed.

Capacity Planning

Similar to dependencies, capacity planning for DNSDB is largely based on the client software integrating DNSDB. Capacity planning is determined by your use cases.

Additional Considerations:

- There are no DNSDB API user-side storage requirements, except to the extent that you elect to save results.
- Conducting right-hand side wildcard RRname searches may be resource intensive.
- When working with large result sets, up to the million results limit per query, adequate RAM must be available.
- Opening more than ten concurrent connections per key per IP address will cause service interruption and/or performance issues. Therefore, some client configuration, IP, and network planning may be required if more than 10 concurrent connections are needed. For mutiple users or systems, depending on the configuration, you may want to request a unique user name and key for each user.

Refer to the capacity requirements for your associated software or platform and make sure there is enough additional CPU, RAM, and storage resources to run DNSDB queries and analyze the results.

Limitations

The following are limitations with DNSDB API:

- By default, DNSDB API users are limited to 10 concurrent connections.
- The maximum number of results you can request is 1,000,000 (one million).
- Partial label queries are not supported; a search has to be done within a domain (e.g. *.example.com) or for a complete domain component (e.g. example.*). Wildcards inside a domain are not supported (can't search for exam*ple.com)
- The order in which DNSDB API returns results does **not** necessarily mean you are being shown the most recent results first by default. You should keep this in mind if you receive only a subset of results, as when results are limited to no more than results.
- Farsight applies significant effort to remove non-authoritative answers from DNSDB. However, it is possible that something was missed by filters, this has the potential to result in invalid data. DNSDB is based on data seen by our sensor nodes and data from zone files. While we will normally see most DNS records in general use, from time to time there will be DNS records a DNSDB sensor has **not** seen. Some low value or privacy sensitive records are also intentionally filtered.

DNSDB Company-provided Demo Command Line Clients

Company-provided demo command line clients are reference implementations of the DNSDB HTTP API. Output is compliant with the Passive DNS Common Output Format. Demo command-line-interface clients are provided in dnsdbq and Python.

Implementation	Description	Required Prerequisites	Additional Information
dnsdbq (formerly knowns as the dnsdb_query C client)	dnsdbq is one of the most feature-rich command line interfaces available for use with DNSDB API.	Linux, BSD, macOS; jansson; libcurl	*dnsdbq* (formerly known as the dnsdb_query C client) - https://github.com/dnsdb/dnsdbq
dnsdb_query.py (Python)	dnsdb_query.py is a Python client for the DNSDB HTTP API. It supports features such as sorting and setting the result limit parameter. It is also embeddable as a Python module.	Linux, BSD, macOS; Curl; Python 2.7.x	*dnsdb_query.py* - https://github.com/dnsdb/dnsdb-query

Third Party Software Integration (Using the DNSDB API)

By augmenting an organization's internal log data with real-time Internet DNS information, security teams will be better able to analyze threats and adversary infrastructure and capabilities. This will enable them to identify, detect, correlate and take action on the intelligence.

Farsight's DNSDB third party software integrations currently include:

- Splunk App (Written by Farsight)—*Farsight DNSDB App for Splunk User Guide* - https://www.farsightsecurity.com/assets/media/download/FarsightSplunkAppUserGuide.pdf
- Maltego Local Transforms—There are Maltego Transforms for DNSDB in the Paterva Transform Hub - *Paterva Transform Hub*- https://www.paterva.com/web7/about/hub.php
- DomainTools Iris—DNSDB is highlighted in the Domain Tools IRIS 2.0 platform
- CyberSponse
- ThreatConnect
- Anomali—In-depth pDNS data is available with the integration of a Farsight API key through Anomali's app store
- ThreatQuotient
- Phantom
- Polarity—*Real-time Collaboration, Real-time Investigation with Farsight DNSDB and Polarity* - https://www.farsightsecurity.com/2018/02/27/kburke-Polarity/
- Swimlane
- Demisto Enterprise
- Siemplify
- Recorded Future
- EcelcticIQ
- ProSoft Systems
- Malware Information Sharing Platform (MISP)—*Farsight DNSDB Now Available on Malware Information Sharing Platform (MISP)* - https://www.farsightsecurity.com/2017/12/12/kburke-MISP/

More integrations are being built every day. Check with your vendor or refer to *Farsight Security Partners With Security Platform Providers* - https://www.farsightsecurity.com/about-farsight-security/partners/

Application Examples

1. Using libcurl to make calls to the DNSDB API

Access *Making Programmatic DNSDB Queries With libcurl* - https://www.farsightsecurity.com/2016/11/04/stsauver-dnsdb-libcurl/

2. Building a Demo GUI Front End for DNSDB API in Scala

Access the whitepaper, *Building a Demo GUI Front End for DNSDB API In Scala with Swing for The Mac and for Windows PCs: A Farsight Security, Inc. Whitepaper* - https://www.farsightsecurity.com/assets/media/download/scala-demo-whitepaper.pdf

3. Checking DNSDB by ASN

Access *Checking DNSDB By ASN (ASN –> Prefixes –> Domain Names) Another Demonstraton Scala Project* - https://www.farsightsecurity.com/assets/media/download/scala-demo-whitepaper2.pdf

Additional References

- *Farsight DNSDB API Documentation* - https://api.dnsdb.info/
- *ISC Passive DNS Architecture* - https://www.farsightsecurity.com/assets/media/download/passive-dns-architecture.pdf
- *DNSDB API New Features* - https://www.farsightsecurity.com/2015/01/14/stern-dnsdbapi/

About Farsight Security

Farsight Security provides the world's largest real-time actionable threat intelligence on changes to the Internet. Leveraging proprietary technology with more than 200,000 observations/second, Farsight provides security teams with the Internet's view of an organization's presence and how it is changing - whether purposely, inadvertently or maliciously. The world's most security-conscious organizations use Farsight threat intelligence to protect their users and infrastructure.

Visit Our Website

https://www.farsightsecurity.com/

Contact Us

Main Telephone: +1-650-489-7919 ; Toll Free: +1-855-489-7919

Farsight Headquarters

Farsight Security, Incorporated 177 Bovet Rd, Suite 180 San Mateo, CA 94402 USA