Workgroup: Domain Name System Operations
Published: 27 August 2024
Intended Status: Informational
Expires: 28 February 2025
Authors: A. Dulaunoy, *CIRCL*; A. Kaplan; P. Vixie, *Farsight Security, Inc.*; H. Stern, *Farsight Security, Inc.*; W. Kumari, *Google*

Passive DNS - Common Output Format

Abstract

This document describes a common output format of Passive DNS servers that clients can query. The output format description also includes a common semantic for each Passive DNS system. By having multiple Passive DNS Systems adhere to the same output format for queries, users of multiple Passive DNS servers will be able to combine result sets easily.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 February 2025.

Copyright Notice

Table of Contents

1. Introduction

Passive DNS is a technique described by Florian Weimer in 2005 in Passive DNS replication, F Weimer - 17th Annual FIRST Conference on Computer Security [WEIMERPDNS]. It is a mechanism for logging DNS answers in a manner intended to minimize the privacy implications to users, and is widely by security researchers to investigate malware (for example to discover command and control servers), and other security threats. By capturing only the "cache fill" DNS responses (responses from authoritative servers in response to queries performed by a recursive resolver when iteratively resolving a name), Passive DNS does not have access to the client (users) source IP, source port, destination IP, or destination port.

As these answers are served in response to queries originally initiated by user devices, the Passive DNS data can be used to detect if devices using the resolver are connecting to known malicious domains, without identifying the individual users / devices. In addition, as answers are responses to queries made by the recursive server itself, Passive DNS records the answers which are ultimately served to users. This is important as authoritative servers may serve different answers to different query addresses, for example to increase performance (e.g Client Subnet in DNS Queries [RFC7871]) or to hide malicious behavior when queried from addresses known to be associated with security researchers.

Passive DNS is usually implemented either by capturing DNS response packets themselves (i.e packets with a destination address of the recursive resolver, a source port of 53, and the QR bit set to 1) or by having the DNS software itself log these responses. The latter method is likely to become more common as recursive to authoritative DNS communication becomes encrypted.

Multiple Passive DNS implementations and services exist. Users of these Passive DNS services may query a server (often via WHOIS [RFC3912] or HTTP REST [REST]), parse the results, and process them in other applications. Users of Passive DNS query each implementation and aggregate the results for their search. This document describes the output format of four Passive DNS Systems ([DNSDB], [DNSDBQ] , [PDNSCERTAT], [PDNSCIRCL] and [PDNSCOF]) that are in use today and that already share a nearly identical output format. As the format and the meaning of output fields from each Passive DNS need to be consistent, this document proposes a solution to commonly name each field along with its corresponding interpretation. The format follows a simple key-value structure in JSON [RFC4627] format. The benefit of having a consistent Passive

DNS output format is that multiple client implementations can query different servers without having to have a separate parser for each individual server. passivedns-client [PDNSCLIENT] currently implements multiple parsers due to a lack of standardization. The document does not describe the protocol (e.g. WHOIS [RFC3912], HTTP REST [REST]) nor the query format used to query the Passive DNS. Neither does this document describe "pre-recursor" Passive DNS Systems. Each of these are separate topics and deserve their own RFC documents. This document describes the current best practices implemented in various Passive DNS server implementations.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Limitations

As Passive DNS servers can include protection mechanisms for their operation, results might be different due to those protection measures. These mechanisms filter out DNS answers if they fail some criteria. The bailiwick algorithm [BAILIWICK] protects the Passive DNS Database from cache poisoning attacks. Another limitation that clients querying the database need to be aware of is that each query simply gets a snapshot-in-time answer at the time of querying. Clients MUST NOT rely on existing answers from different Passive DNS database. Nor should they assume that answers will be identical across multiple Passive DNS servers.

3. Common Output Format

3.1. Overview

The formatting of the answer follows the JSON [RFC4627] format. In fact, it is a subset of the full JSON language. Notable differences are the modified definition of whitespace ("ws"). The order of the fields is not significant for the same resource type.

The intent of this output format is to be easily parsable by scripts. Each JSON object is expressed on a single line to be processed by the client line-by-line. Every implementation MUST support the JSON output format.

Examples of JSON (Appendix A) output are in the appendix.

3.2. ABNF grammar

Formal grammar as defined in ABNF [RFC2234]

```
answer         = entries
entries        =  *  ( entry newline )
entry          = ws "{" ws keyvallist ws "}" ws
keyvallist     = [ member  *  ( value-separator member ) ]
member         = field name-separator value
name-separator  = ws %x 3 A ws          ; : colon
value-separator = ws %x 2 C ws          ; , comma
field          = field-name | futureField
field-name     = "rrname" | "rrtype" | "rdata" | "time_first" |
                 "time_last" | "count" | "bailiwick" | "sensor_id" |
                 "zone_time_first" | "zone_time_last" | "origin" |
                 "time_first_ms" | "time_last_ms"
futureField    = string
newline        = [ CR ] LF
CR             = %x 0 D                 ; Carrige return
LF             = %x 0 A                 ; Line feed or New line
qm             = %x 2  2                  ; " Quotation mark
ws             =  *  (
                  %x 2  0  |            ; Space
                  %x 0  9              ; Horizontal tab
                  )
```

Figure 1

Note that value is defined in JSON [RFC4627] and has the same specification as there. The same goes for the definition of string. Note the changed definition of ws does not include CR or LF as those are NOT allowed in NDJSON, and thus the definition here MUST be used for other ABNF defitions in JSON [RFC4627] .

3.3. Mandatory Fields

Implementation MUST support all the mandatory fields.

Uniqueness property: the tuple (rrname,rrtype,rdata) will always be unique within one answer per server. While rrname and rrtype are always individual JSON primitive types (strings, numbers, booleans or null), rdata MAY return multiple resource records or a single record. When multiple resource records are returned, rdata MUST be a JSON array. In the case of a single resource record is returned, rdata MUST be a JSON string or a JSON array containing one JSON string. Senders SHOULD send an array for rdata, but receivers MUST be able to accept a single-string result for rdata.

3.3.1. rrname

This field returns the name of the queried resource. Represented as a JSON [RFC4627] string.

3.3.2. rrtype

This field returns the resource record type as seen by the passive DNS. The key is rrtype and the value is in the interpreted record type represented as a JSON [RFC4627] string. If the value cannot be interpreted, the decimal value is returned, following the principle of transparency as described in RFC 3597 [RFC3597]. Then the decimal value is represented as a JSON [RFC4627] number. The resource record type can be any values as described by IANA in the DNS parameters document in the section 'Resource Record (RR) TYPEs' (http://www.iana.org/assignments/dns-parameters). Supported textual descriptions of rrtypes include: A, AAAA, CNAME, etc. A client MUST be able to understand these textual rrtype values represented as a JSON [RFC4627] string. In addition, a client MUST be able to handle a decimal value (as mentioned above) answer represented as a JSON [RFC4627] number.

3.3.3. rdata

This field returns the resource records of the queried resource. When multiple resource records are returned, rdata MUST be a JSON array containing JSON strings. In the case of a single resource record being returned, rdata MUST be a JSON string or a JSON array containing one JSON string. Each resource record is represented as a JSON [RFC4627] string. Each resource record MUST be escaped as defined in section 2.6 of RFC4627 [RFC4627]. Depending on the rrtype, this can be an IPv4 or IPv6 address, a domain name (as in the case of CNAMEs), an SPF record, etc. A client MUST be able to interpret any value which is legal as the right hand side in a DNS master file RFC 1035 [RFC1035] and RFC 1034 [RFC1034]. If the rdata came from an unknown DNS resource records, the server must follow the transparency principle as described in RFC 3597 [RFC3597] .

3.3.4. time_first

This field returns the first time that the record / unique tuple (rrname, rrtype, rdata) has been seen by the passive DNS. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.3.5. time_last

This field returns the last time that the unique tuple (rrname, rrtype, rdata) record has been seen by the passive DNS. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.4. Optional Fields

Implementations SHOULD support one or more fields.

3.4.1. count

Specifies how many authoritative DNS answers were received at the Passive DNS server's collectors with exactly the given set of values as answers (i.e. same data in the answer set - compare with the uniqueness property in "Mandatory Fields"). The number of requests is expressed as a decimal value. This field is represented as a JSON [RFC4627] number.

3.4.2. bailiwick

The bailiwick is the best estimate of the apex of the zone where this data is authoritative. This field is represented as a JSON [RFC4627] string.

3.5. Additional Fields

Implementations MAY support the following fields:

3.5.1. sensor_id

This field returns the sensor information where the record was seen. It is represented as a JSON [RFC4627] string.

If the data originate from sensors or probes which are part of a publicly-known gathering or measurement system (e.g. RIPE Atlas), a JSON [RFC4627] string SHOULD be prefixed.

3.5.2. zone_time_first

This field returns the first time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.5.3. zone_time_last

This field returns the last time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.5.4. origin

Specifies the resource origin of the Passive DNS response. This field is represented as a Uniform Resource Identifier [RFC3986] (URI) in the form of a JSON [RFC4627] string.

3.5.5. time_first_ms

Same meaning as the field "time_first", with the only difference, that the resolution is in milliseconds since 1st of January 1970 (UTC).

3.5.6. time_last_ms

Same meaning as the field "time_last", with the only difference, that the resolution is in milliseconds since 1st of January 1970 (UTC).

3.6. Additional Fields Registry

In accordance with [RFC6648], designers of new passive DNS applications that would need additional fields can request and register new field name at https://github.com/adulau/pdns-qof/wiki/Additional-Fields.

3.7. Additional notes

An implementer of a passive DNS server MAY chose to either return time_first and time_last OR return zone_time_first and zone_time_last. In pseudocode: (time_first AND time_last) OR (zone_time_first AND zone_time_last). In this case, zone_time_{first,last} replace the time_{first,last} fields. However, this is not encouraged since it might be confusing for parsers who will expect the mandatory fields time_{first,last}. See: [github_issue_17]

3.8. Suggested MIME Types

An implementer of a passive DNS server SHOULD serve a document in this Common Output Format with a MIME header of "application/x-ndjson".

4. Acknowledgements

Thanks to the Passive DNS developers who contributed to the document.

5. IANA Considerations

This memo includes no request to IANA.

6. Privacy Considerations

Passive DNS servers capture DNS answers from multiple collection points ("sensors") which are located on the Internet-facing side of DNS recursors ("post-recursor passive DNS"). In this process, they intentionally omit the source IP, source port, destination IP and destination port from the captured packets. Since the data is captured "post-recursor", the timing information (who queries what) is lost, since the recursor will cache the results. Furthermore, since multiple sensors feed into a passive DNS system, the resulting data gets mixed together, reducing the likelihood that Passive DNS systems are able to find out much about the actual person querying the DNS records. In this sense, passive DNS systems are similar to keeping an archive of all previous phone books - if public DNS records can be compared to phone numbers - as they often are. Nevertheless, the authors strongly encourage Passive DNS implementors to take special care of privacy issues. Finally, the overall recommendations in RFC6973 [RFC6973] should be taken into consideration when designing any application which uses Passive DNS data.

Passive DNS attempts to collect information necessary for security (such as malware protection) in as privacy protecting a manner as possible, and is intended to be used instead of more invasive methods. It does this by only collecting DNS cache-fill answers, and not any information associated with who caused the name to be resolved, nor why the name was resolved. Nevertheless, it is possible that this may still lead to privacy concerns - for example, if Passive DNS records show that a recursive resolver resolved the name the-mary-and-john-smith-

family.example.com, it may be possible to infer that the Smith family is using that resolver. Operators of Passive DNS servers should be aware of this and take appropriate steps to limit access to the data.

Passive DNS operators are encouraged to read and understand RFC7258 [RFC7258]

In the scope of the General Data Protection Regulation (GDPR - Directive 95/46/EC), operators of Passive DNS server needs to ensure the legal ground and lawfulness of its operation.

7. Security Considerations

In some cases, Passive DNS output might contain confidential information and its access should be restricted. When a user is querying multiple Passive DNS and aggregating the data, the sensitivity of the data must be considered.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <https://www.rfc-editor.org/info/rfc4627>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/info/rfc6648>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[WEIMERPDNS] Weimer, F., "Passive DNS Replication", 2005, <http://www.enyo.de/fw/software/dnslogger/first2005-paper.pdf>.

[PDNSCOF] Dulaunoy, D. P. A., "Passive DNS server interface using the common output format", 2019, <https://github.com/D4-project/analyzer-d4-passivedns/>.

[github_issue_17] et.al, P. V. W. A. K., "Discussion on the existing implementations of returning either zone_time{first,last} OR time_{first,last}", 2020, <https://github.com/adulau/pdns-qof/issues/17>.

9. Informative References

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[BAILIWICK] Edmonds, R., "Passive DNS Hardening", 2010, <https://archive.farsightsecurity.com/Passive_DNS/passive_dns_hardening_handout.pdf>.

[PDNSCLIENT] Lee, C., "Queries 5 major Passive DNS databases: BFK, CERTEE, DNSParse, ISC, and VirusTotal.", 2013, <https://github.com/chrislee35/passivedns-client>.

[REST] Fielding, R. T., "Representational State Transfer (REST)", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.

[DNSDB] Security, F., "DNSDB API", 2013, <https://api.dnsdb.info/>.

[PDNSCERTAT] CERT.at, "pDNS presentation at 4th Centr R&D workshop Frankfurt Jun 5th 2012", 2012, <http://www.centr.org/system/files/agenda/attachment/d4-papst-passive_dns.pdf>.

[PDNSCIRCL] Luxembourg, C. -. I. R. C., "CIRCL Passive DNS", 2012, <https://www.circl.lu/services/passive-dns/>.

[DNSDBQ] Vixie, P., "DNSDB API Client, C Version", 2018, <https://github.com/dnsdb/dnsdbq>.

Appendix A. Examples

The JSON output are represented on multiple lines for readability but each JSON object should be on a single line.

If you query a passive DNS for the rrname www.ietf.org, the passive dns common output format can be:

```
{"count": 102, "time_first": 1298412391, "rrtype": "AAAA",
"rrname": "www.ietf.org", "rdata": "2001:1890:1112:1::20",
"time_last": 1302506851}
{"count": 59, "time_first": 1384865833, "rrtype": "A",
"rrname": "www.ietf.org", "rdata": "4.31.198.44",
"time_last": 1389022219}
```

Figure 2

If you query a passive DNS for the rrname ietf.org, the passive dns common output format can be:

```
{"count": 10987707, "time_first": 1298398300, "rrtype": "NS",
"rrname": "ietf.org", "rdata": "ns1.yyz1.afilias-nst.info",
"time_last": 1389095375}
{"count": 4, "time_first": 1298429504, "rrtype": "A",
"rrname": "ietf.org", "rdata": "64.170.98.32",
"time_last": 1298429504}
{"count": 9, "time_first": 1317037550, "rrtype": "AAAA",
"rrname": "ietf.org", "rdata": "2001:1890:123a::1:e",
"time_last": 1330209752}
```

Figure 3

Please note that the examples imply that a single query returns a single set of JSON objects. For example, two queries were made; one query returned a set of two JSON objects and the other query returned a set of three JSON objects. This specification requires each JSON object individually MUST conform to the common output format, but this specification does not require that a query will return a set of JSON objects.

Please note that in the examples above, any backslashes "\" can be ignored and are an artifact of the tools which produced this document.

Authors' Addresses

Alexandre Dulaunoy
CIRCL
122, rue Adolphe Fischer
L-L-1521 Luxembourg
Luxembourg
Phone: (+352) 247 88444
Email: alexandre.dulaunoy@circl.lu
URI: http://www.circl.lu/

L. Aaron Kaplan
A-1170 Vienna
Austria
Email: aaron@lo-res.org

Paul Vixie
Farsight Security, Inc.
11400 La Honda Road
Woodside, California 94062
United States of America
Email: paul@redbarn.org
URI: https://www.farsightsecurity.com/

Henry Stern
Farsight Security, Inc.
11400 La Honda Road
Woodside, California 94062
United States of America
Phone: +1 650 542-7836
Email: henry@stern.ca
URI: https://www.farsightsecurity.com/

Warren Kumari
Google
Email: warren@kumari.net