.SE (The Internet Infrastructure Foundation)

DNS^{2db} User's manual

v2.2.1

January 2010

1 Preface

This user's manual has been written to help those who decide to run the DNS^{2db} system. More information about building or installing DNS^{2db} and developer information such as an xml-format specification and an interface specification of the php files can be found at:

`http://opensource.iis.se/trac/dns2db/`

The man pages for tracedns and dns2sqlite may be useful for those who would like to customize the perl job control daemon or roll their own. Information about third party software used in DNS^{2db} can be found at the following locations.

libtrace	http://research.wand.net.nz/software/libtrace.php
ldns	http://www.nlnetlabs.nl/ldns/
SQLite3	http://www.sqlite.org/
Adobe Flex3	http://www.adobe.com/products/flex/

Contents

2 Introduction

DNS^{2db} is a DNS query statistics and analysis application.

It is used to monitor and analyze traffic on DNS servers, especially during traffic peaks where it is helpful to know the cause of the peak. It may also be used to identify serious errors in DNS configuration which causes unnatural DNS traffic.

The design of DNS^{2db} makes it possible to get this information within a few minutes and could be extended to report near realtime information on for example load and various predefined patterns. It may also be used to do statistical analysis on DNS traffic over time.

3 Overview

DNS^{2db} consists of two main parts, a collector backend and a GUI. The backend collects and converts raw pcap data from a network into sqlite3 databases. Collecting and converting is done in two separate processes, tracedns for capturing and dns2sqlite for parsing and storing to database. If there is a need to store pcap data for later reference this can best be done by capturing pcap to disk before conversion. The data collecting is done in close proximity to the dns server and the collected information is made accessible to the user through a web based GUI using the XML API.

The GUI is written as an Adobe Flex application that runs in any flash enabled web browser. The Flex GUI can get information from the databases through a php script that specifies and limits the information available to the GUI. The GUI application is served completely from the server and no additional software is needed on the client. Figure 1 is a description of a complete DNS^{2db} implementation on a separate packet capturing server that listens on incoming and outgoing traffic to a DNS server. The setup for sniffing traffic can be done with port forwarding in a connected switch, with a transparent ethernet splitter or by other means. It can also be done on the DNS server itself but this is not recommended. The DNS^{2db} system is started and stopped by a wrapper script named dns2db.

4 The Collector nodes

4.1 Overview

The following files are part of a DNS^{2db} collector node:

File	Description
`tracedns`	The input module of DNS^{2db}
`dns2sqlite`	The output module of DNS^{2db}
`/etc/dns2db.conf`	Configuration file
`/etc/init.d/dns2db`	Init script
`/usr/bin/dns2db.pl`	Process control script (used by /etc/init.d/dns2db)
`dns2dbnode.php`	php backend on each collector node
`dns2dbnode_conf.php`	php config file for the collector node

The dns2db init script reads configuration parameters in /etc/dns2db.conf and calls /usr/bin/dns2db.pl to start the tracedns and the dns2sqlite binary with command line arguments from the dns2db.conf file. Arguments to the wrapper script are start|stop. Configuration parameters for the processes are set by the wrapper during startup and cannot be changed during run time. Optionally the start script starts a separate capturing process to store pcap data and runs the DNS^{2db} processes after every new pcap file. The tracedns process reads from the configured input and writes to stdout. Input can be any file or interface that libtrace can handle. Eg "pcapint:eth0" for the first network interface on a Linux system or "pcapfile:/a-file.pcap" for a pcap file on a local drive. The dns2sqlite process listens to stdin and creates the databases in the path set in dns2db.conf.

4.2 The dns2db.pl job control daemon

The dns2db.pl job control daemon is usually started by the initscript dns2db and is the process that starts tracesplit to generate pcap files and then tracedns and dns2sqlite to generate sqlite databases for each interval. It also indexes the databases. The dns2db.pl script is configure by /etc/dns2db.conf. Configuration parameters for the processes are set up during startup and cannot currently be changed during run time.

4.3 Initscript

The dns2db init script reads configuration parameters in /etc/dns2db.conf and calls /usr/bin/dns2db.pl which starts the tracedns and the dns2sqlite binary with command line arguments from the dns2db.conf file. Arguments to the wrapper script are start|stop. This script may need to be customized to your particular operating system.

4.4 The tracedns command

The tracedns process reads from the configured input and writes to stdout. Input can be any file or interface that libtrace can handle. Eg pcapint:eth0 for the first network interface on a Linux system or pcapfile:/a-file.pcap for a pcap file on a local drive.

4.5 The dns2sqlite command

The dns2sqlite process listens to stdin and creates the databases in the path set in dns2db.conf.

4.6 The configuration file

Heres an example configuration file (/etc/dns2db.conf).

```
## dns2db pid file
pidfile=/var/run/dns2db.pid

## Name of the DNS server. Parameter is used first in filename
## when creating tcpdump files and sql files.
server="servername"

## Ramdisk where database is created and indexed
workdir=/tmp/workdir

## Final directory where databases and pcap files are stored
# make sure path ends with trailing "/"
destdir=/tmp/outputdir/

## chmod finished files to serve from apache
user=www-apache-httpd

## Name of the network interface to monitor
interface=eth0

## How often to rotate dump file, in seconds
interval=300

## Keep pcap data
keeppcap=YES

## Compress pcap data
compresspcap=YES

## BSD libtrace promiscous interface hack
# uses a tcpdump session on port 100 to keep the interface
# in promisc mode because tracesplit seems to be unable
# to do so on bsd
bsdpromischack=NO

## path to the tcpdump binary
tcpdump=tcpdump

## path to the tracesplit binary
# tracesplit is distributed in the tools folder of the libtrace
# library make sure it's built and installed.
tracesplit=/usr/local/bin/tracesplit

### choose a packet filter:
## collect TCP and UDP, requests and responses:
filter="port 53"

## create sqlite index
index="create index ix_src_addr on q (src_addr);create index
domain on q (rr_lvl2dom,rr_lvl1dom);create index ix_rr_type
on q (rr_type); create index if not exists resanddom on q
(rr_cname,src_addr,rr_type);"
```

The fields that must be changed to have a working collector are workdir, destdir, user and interface.

- The parameter **workdir** specifies a working directory for optimal performance this should be placed on a ramdisk with enough space for pcap files and database files for three peek intervals.
- The **destdir** path points to the path where the database files are to be stored.
- The **user** parameter specifies the user that will own the resulting files. This should normally be the web server user to allow the databases to be accessed by the php scripts.
- The **interface** parameter specifies which network interface to listen on i.e. eth0.
- The **filter** parameter can be used to specify a berkely packet filter string to trancedns that can be used to filter the output to specific ports and/or hosts. This is usually "port 53" to only see dns traffic but could also include filters to i.e. filter out dns queries from the dns server itself ("port 53 and not host 192.168.0.1").

4.7 The PHP scripts

The file dns2db_node.php serves results through a web server to dns2db.php on the gui server which may or may not be the same machine.

Normally the dns2db_node.php script uses the database path from /etc/dns2db.conf but in cases where the web server is running chrooted it's recommended to rename the file dns2dbnode_conf.php.example into dns2dbnode_conf.php and to change the path within to correspond to the destdir of /etc/dns2db.conf.

5 The gui server

The following files are part of the GUI server:

File	Description
index.php	html part of the Adobe Flex application
dns2db.swf	the Adobe Flex application
dns2db.php	php backend for the Flex application
dns2db_conf.php	config file for the GUI
reversedb.db3	reverse lookup cache for the GUI

The GUI consists of a flash application, a set of php scripts and a reverse DNS lookup cache database. When the flash application is loaded in the users webbrowser it will default to presenting toplists for current time in UTC minus 5 minutes.

Note: reversedb.db3 must be writable by the web server user, also applies to the catalogue containing the reversedb.db3.

5.1 The PHP configuration file

The file dns2db_conf.php is the only file that needs to be changed on the GUI server. The default file looks like this:

```
1  <?php
2  ########## urls for the swf and dns2db.php files
3  #
4  #  the default relative paths below should be fine for most users
5  $swf = "dns2db.swf";
6  $url = "dns2db.php";
7
8
9  ########## reverse lookup cache
10 # this specifies the location of the name lookup cache database
11 # make sure both this file and the directory it's in is
12 # writable by the web server user
13 $database = "reversedb.db3";
14
15 $nodes = array (  # start nodelist
16
17 ##########  nodelist   ############################################################
18 # add each collector node to the $nodes array here using a
19 # line like the example below
20 # make sure the 'url' is reachable by the dns2db script and that
21 # the 'name' is free of any whitespace characters
22 #
23     array(  'name' => 'x' ,
24             'dnsname' => 'x.example.comm',
25             'url' => 'http://x.example.comm/dns2db/test/dns2dbnode.php',
26             'displayname'=>'X',
27             'description'=>'the server X'),
28
29 #   array(  'name' => 'y' ,
30             'dnsname' => 'y.example.comm',
31             'url' => 'http://y.example.comm/dns2db/dns2dbnode.php',
32             'displayname'=>'Y',
33             'description'=>'the server Y'),
34
35 );         # end nodelist
36
```

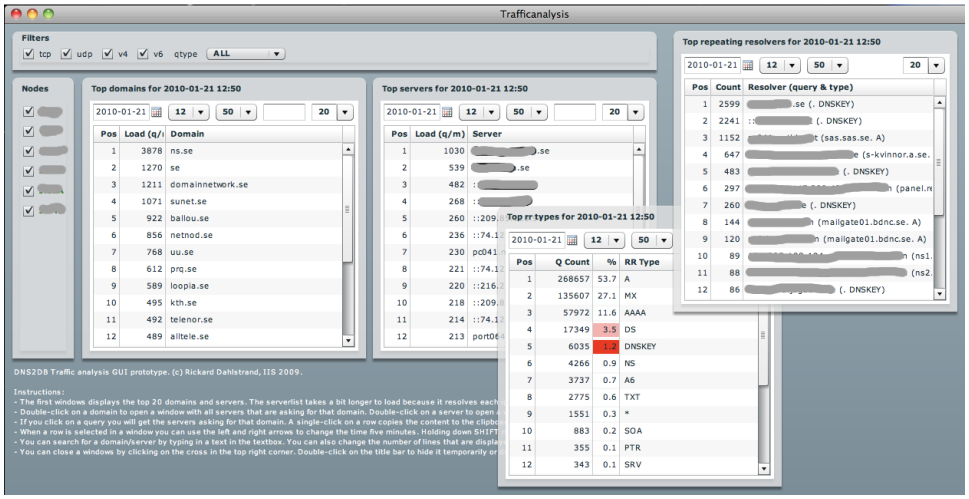
```
37
38 $rrstats = array (  # start rrstats
39 ##########  rrstats   ############################################################
40 #
41 #  this array provides data for coloring the rrtypes display in the gui
42 #  deviation controls the deviation where coloring occurs
43
44    array( 'name' => 'A' , 'percent' => '49', 'deviation' => '4'),
45    array( 'name' => 'MX' , 'percent' => '23', 'deviation' => '3'),
46    array( 'name' => 'AAAA' , 'percent' => '13', 'deviation' => '2'),
47    array( 'name' => 'NS' , 'percent' => '10', 'deviation' => '2'),
48    array( 'name' => 'TXT' , 'percent' => '1', 'deviation' => '0.5'),
49    array( 'name' => 'DS' , 'percent' => '0.9', 'deviation' => '0.5'),
50    array( 'name' => 'A6' , 'percent' => '0.8', 'deviation' => '0.5'),
51    array( 'name' => '*' , 'percent' => '0.4', 'deviation' => '.3'),
52    array( 'name' => 'SOA' , 'percent' => '0.2', 'deviation' => '.2'),
53    array( 'name' => 'SRV' , 'percent' => '0.1', 'deviation' => '0.1'),
54    array( 'name' => 'SPF' , 'percent' => '0.1', 'deviation' => '0.1'),
55    array( 'name' => 'PTR' , 'percent' => '0.1', 'deviation' => '0.1'),
56    array( 'name' => 'CNAME' , 'percent' => '0.1', 'deviation' => '0.1'),
57    array( 'name' => 'DNSKEY' , 'percent' => '0.1', 'deviation' => '0.1'),
58    array( 'name' => 'RRSIG' , 'percent' => '0.1', 'deviation' => '0.1'),
59    array( 'name' => 'NSEC' , 'percent' => '0.1', 'deviation' => '0.1'),
60
61 );      # end rrstats
62 ?>
```

The most important part of this configuration file is the nodelist where you configure you nodes. Each node needs to have a name,dnsname,url,displayname and description.

- **name** specifies a short unique name for the node used in communication between the flex gui client and the gui server.
- **dnsname** specifies the dnsname of the node this is not currently used in the gui.
- **url** specifies the http url where the nodes dns2dbnode.php script can be accessed by the gui server.
- **displayname** Is the short string next to each node that is displayed on the left edge of the gui.
- **description** Should contain a short description of the node. Not currently used by the gui but this string is intended to be used as a tooltip for the node.

The rrstats array decides the coloring in the top rr types in the gui this is not essential to the operation but these numbers should be set so that the **percent** number plus minus the **deviation** number covers normal usage values as displayed in the **Top rr types** window in the flexgui (see next chapter). When usage is outside these numbers the fields will be colored red to draw attention to those querytypes.

6 The flash user interface



The flash user interface consists of a number of windows that show information or control which information to show about the usage of the nodes. To use the flash user interface you need a flash enabled web browser.

6.1 Nodes

The nodes window controls which nodes you currently see output from in the top list windows. If a nodes name is colored green it has delivered data in response to the last request and if the name is colored red it has not.

6.2 Filters

Filters control what protocols and query type are currently shown. Deselecting i.e. udp and v4 will cause udp packets and ip v4 packets to be filtered out and only tcp packets over ip v6 will be shown.

6.3 The list windows

Common for the list windows is that they have a date, time, a filter box and a number of results field. When a window is selected you can step 5 minute intervals using the left and right arrow keys. If SHIFT is held down the arrows key will step hours instead and if the CTRL key is held it will step days. The time between the **Top domains**, **Top rr types**, **Top servers** and **Top repeating resolvers** windows are linked and always the same.

Note that loads are expressed as queries per minute and a load of 0 queries per minute may mean anything between 1 and 4 queries during a five minute interval.

6.4 Top domains

The Top domains window show the top domains ordered by queries per minute. Clicking on a domain in the list will open the Servers asking about [domain] window for that resolver.

6.5 Top servers

The Top servers window show the top resolvers ordered by queries per minute. Clicking on a resolver in the list will open the Queries from [server] window for that resolver.

6.6 Top rr types

The Top servers window show the top rr types ordered by queries per minute. The fields are colored according to the rrstats array in the configuration for the gui server.

6.7 Top repeating resolvers

The Top repeating resolvers window show the top resolvers repeating the same query ordered by repeated queries per minute.

6.8 Servers asking about [domain]

The Servers asking about [domain name] window show resolvers asking about a certain domain ordered by queries per minute.

6.9 Queries from [server]

The Queries from [server] window show queries from a certain resolver ordered by queries per minute.