DNS CACHE

一 总述

DNS cache 用来响应恶意dns攻击，当DNS权威服务器受到攻击时，上层路由策略会将超过额定流量的多余请求转发到DNS cache模块，以用来保证DNS权威服务器正常工作, 鉴于DNS cache的特性，所以不需要记录日志

DNS cache只负责接收权威服务器的应答数据或者外部dns请求数据。所以需要一个DNS cache server处理网络数据包， 同时需要一个DNS data store DNS数据库来存放缓存数据。所以DNS cache的功能模块如下图。



二 程序相关结构体定义

struct pkg\_data

{

void \*data;

uint16\_t data\_len;

uint16\_t type;

struct pkg\_data \*next;

};

struct pkg\_list

{

pkg\_data\_t \*header;

uint32\_t count;

uint32\_t mem\_size;

struct pkg\_list \*prev;

struct pkg\_list \*next;

wire\_name\_t \*owner;

};

struct record\_store

{

domain\_rbtree\_t \*domain\_rbtree\_;

lru\_list\_t \*lru\_list\_;

uint32\_t total\_mem\_size;

};

struct lru\_list

{

struct pkg\_list \*header;

struct pkg\_list \*tail;

uint32\_t count;

};

typedef struct query\_session

{

uint8\_t raw\_data[UDP\_MAX\_PACKAGE\_LEN];

uint16\_t data\_len;

socket\_t client\_socket;

addr\_t client\_addr;

wire\_name\_t \*wire\_name;

dnscache\_t \*dnscache;

}query\_session\_t;

struct dnscache

{

struct event\_base \*g\_base\_event\_;

struct event \*g\_sig\_int\_event\_;

struct event \*g\_sig\_term\_event\_;

dns\_server\_t \*g\_dns\_server\_;

struct record\_store \*g\_record\_store\_;

mem\_pool\_t \*mp\_;

};

各模块接口函数

1 dns\_package\_manager

int pkg\_get\_query\_type(const char \*raw\_data, uint16\_t data\_len);

int pkg\_get\_query\_id(const char \*raw\_data, uint16\_t data\_len);

void pkg\_set\_response\_id(char \*pkg\_data, uint32\_t id);

2 record store

record\_store\_t \* record\_store\_create();

record\_store\_delete(record\_store\_t \*record\_store);

void

record\_store\_insert\_pkg(record\_store\_t \*record\_store,

const char \*raw\_data,

const uint16\_t data\_len);

bool

record\_store\_get\_pkg(record\_store\_t \*record\_store,

char \*raw\_data,

uint16\_t \*data\_len);

3 server

static void \*

dnscache\_handle\_query\_pkg(void \*arg)

static void \*

dnscache\_handle\_response\_pkg(void \*arg)

DNS cache 工作流程图如下



2 LRU 列表

为了合理利用系统内存资源，dns cache 采用LRU（latest recently used）进行内存资源的管理。

