目录

抓包

纯真数据库没有提供单独的文件下载，只在 Windows 下提供了一个可以查询和更新数据库的桌面软件；所以想要在服务器上自动更新 IP 数据库，需要对它更新的时候进行抓包，看看它是怎么从服务器端下载新的数据库文件的。

Figure 1: qqwry

使用 wireshark 抓包，filter 过滤表达式如下

```
http.request.method == 'GET' and http.host == 'update.cz88.net'
```

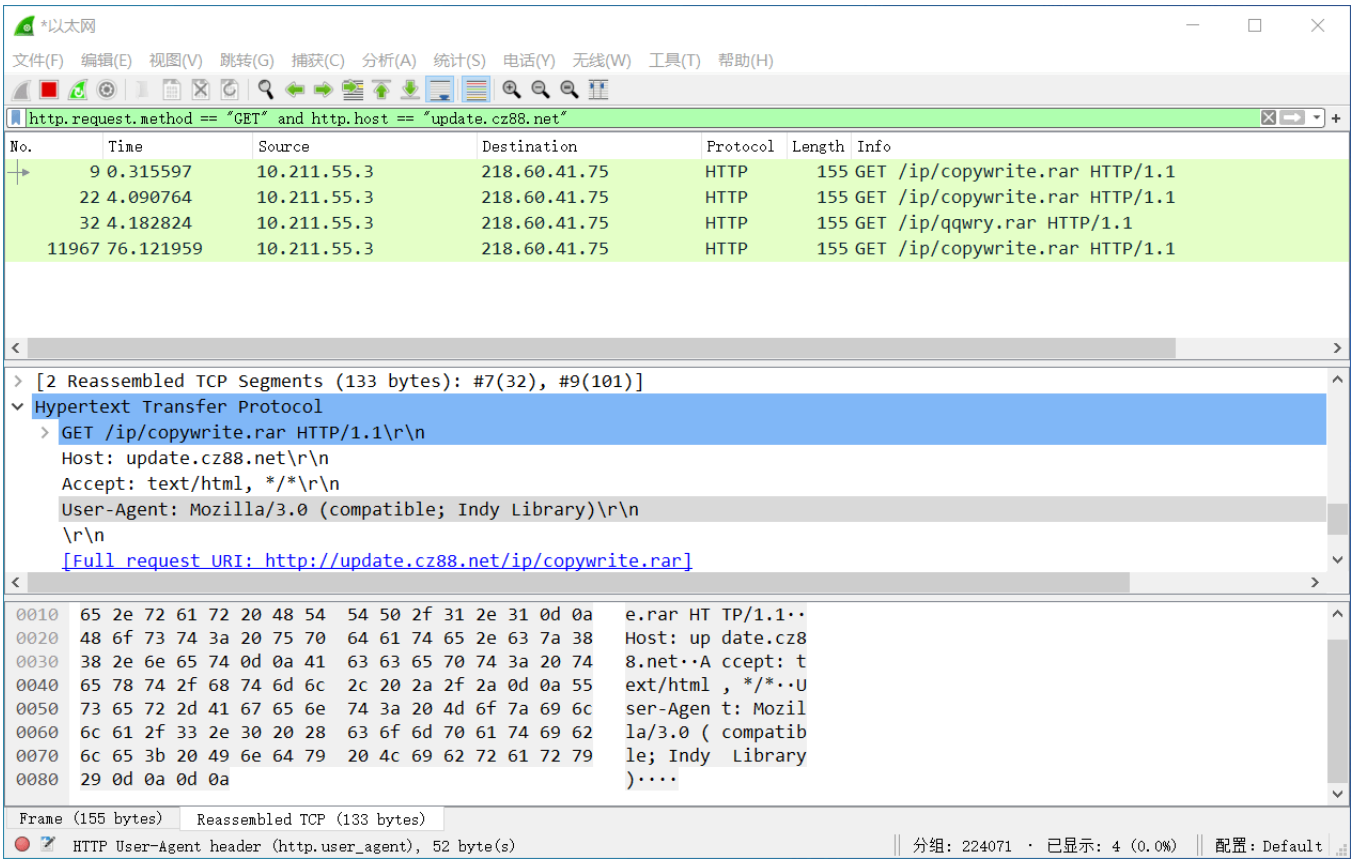

Figure 2: wireshark

抓包发现了两个请求 `/ip/copywrite.rar` 和 `/ip/qqwry.rar`;copy 其中的 http request 头信息,`Accept: text/html, */*` 和 `User-Agent: Mozilla/3.0 (compatible; Indy Library)`，至于为什么要记录这两个头，使用浏览器打开这两个链接就知道了，是没有办法直接打开的，我们需要模拟它的桌面端，发起请求。

文件的结构如下：当我们使用上述的方式将 `copywrite.rar` 和 `qqwry.rar` 下载下来之后，怎么变成 `qqwry.dat` 呢？以 rar 结尾的，但并不是 rar 文件。

> todo IDA pro 分析过程

纯真 IP 数据库

在线工具

copywrite.rar

sign	version	unknown1	size	unknown2	key	text	link
uint32	uint32	uint32	uint32	uint32	uint32	char 128	char 128

Figure 3: copywrite.rar

qqwry.rar 的前 512(0x200) 个字节，需要用到 copywrite.rar 中的 key 进行解密。

自动更新

```
import struct
import zlib
import requests
import logging
import time


logging.getLogger().setLevel(logging.INFO)

class QQWryUpdater(object):
    version_file = "./tmp/qqwry_version.bin"
    tmp_file = "./tmp/qqwry.dat"
    target_file = "./data/qqwry.dat"
    copywrite_url = "http://update.cz88.net/ip/copywrite.rar"
    qqwry_url = "http://update.cz88.net/ip/qqwry.rar"
    headers = {
        "User-Agent": "Mozilla/3.0 (compatible; Indy Library)",
        "Accept": "text/html, */*",
    }

    @classmethod
    def update(cls):
```

```
        curr_version, check_time, update_time = (0, 0, 0)
        with open(cls.version_file, "rb+") as handle:
            content = handle.read()
            if len(content) > 0:
                curr_version, check_time, update_time = struct.unpack("<3I", content)

        check_time = int(time.time())
        copywrite = requests.get(cls.copywrite_url, headers = cls.headers).content
        # copywrite[:4] == b"CZIP" # 纯真 IP

        version, unknown1, size, unknown2, key = struct.unpack_from("<5I", copywrite,
↪  4)
        if version == curr_version:
            logging.info("no update for version %d", curr_version)
            with open(cls.version_file, "wb+") as handle:
                handle.write(struct.pack("<3I", version, check_time, update_time))
            return
        logging.info("new version %d, size %d", version, size)

        update_time = int(time.time())
        qqwry = requests.get(cls.qqwry_url, headers = cls.headers).content

        head = bytearray(0x200)
        for i in range(0x200):
            key = (key * 0x805 + 1) & 0xFF
            head[i] = qqwry[i] ^ key

        qqwry = head + qqwry[0x200:]
        data = zlib.decompress(qqwry)
        with open(cls.tmp_file, "wb") as handle:
            handle.write(data)

        with open(cls.version_file, "wb+") as handle:
            handle.write(struct.pack("<3I", version, check_time, update_time))

    @classmethod
    def cover(cls):
        if os.path.isfile(cls.tmp_file):
            shutil.move(cls.tmp_file, cls.target_file)

    @classmethod
    def updateAndCover(cls):
        cls.update()
        cls.cover()
        logging.info("[DONE]")

if __name__ == '__main__':
    QQWryUpdater.updateAndCover()
```

发布

自动更新完了还没了事，跟发布系统一样；我们的文件不能更新到一半的时候使用，所以需要更新完之后覆盖回去；那就涉及到 mv 和 cp 的选择, 这里我们必须选择 mv，因为 linux 系统中记录的文件的 inode，mv 只是 rename 了这个 inode，没有进行实际的文件 copy