TCPCopy - A TCP Stream Replay Tool

TCPCopy is a TCP stream replay tool to support real testing of Internet server applications.

Description

Although the real live flow is important for the test of Internet server applications, it is hard to simulate it as online environments are too complex. To support more realistic testing of Internet server applications, we develop a live flow reproduction tool - TCPCopy, which could generate the test workload that is similar to the production workload. Currently, TCPCopy has been widely used by companies in China.

TCPCopy has little influence on the production system except occupying additional CPU, memory and bandwidth. Moreover, the reproduced workload is similar to the production workload in request diversity, network latency and resource occupation.

Scenarios

- Distributed stress testing

    - Use tcpcopy to copy real-world data to stress test your server software. Bugs that only can be produced in high-stress situations can be found
- Live testing
    - Prove the new system is stable and find bugs that only occur in the real world
- Regression testing
- Performance comparison

Architecture

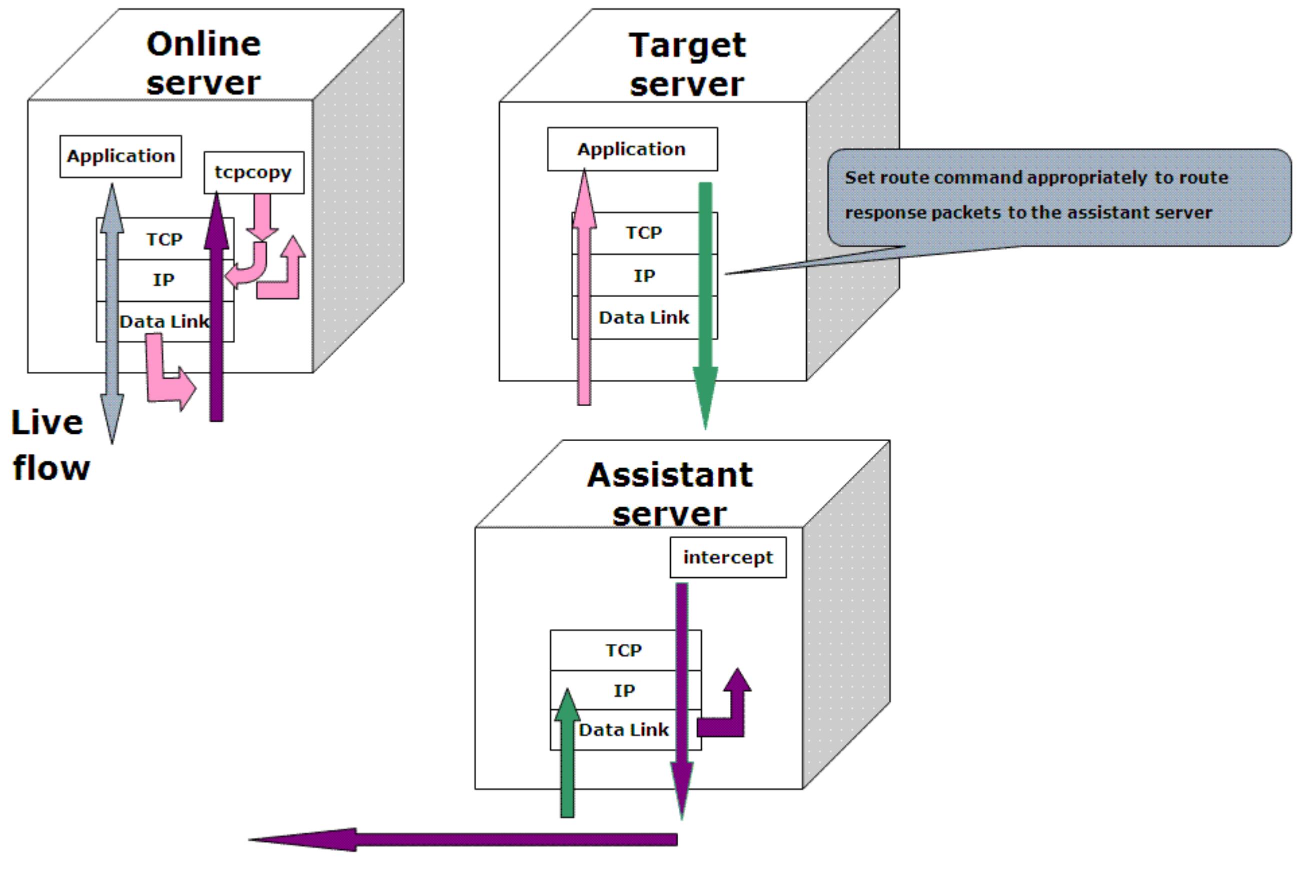

Figure 1

As shown in Figure 1, TCPCopy consists of two parts: *tcpcopy* and *intercept*. While *tcpcopy* runs on the online server and captures the online requests, *intercept* runs on the assistant server and does some assistant work, such as passing response info to *tcpcopy*. It should be noted that the test application runs on the target server.

tcpcopy utilizes raw socket input technique by default to capture the online packets at the network layer and does the necessary processing (including TCP interaction simulation, network latency control, and common upper-layer interaction simulation), and uses raw socket output technique by default to send packets to the target server (shown by pink arrows in the figure).

The only operation needed on the target server for TCPCopy is setting appropriate route commands to route response packets (shown by green arrows in the figure) to the assistant server.

intercept is responsible for passing the response header(by default) to *tcpcopy*. By capturing the response packets, *intercept* will extract response header information and send the response header to *tcpcopy* using a special channel(shown by purple arrows in the figure). When *tcpcopy* receives the response header, it utilizes the header information to modify the attributes of online packets and continues to send another packet. It should be noticed that the responses from the target server are routed to the assistant server which should act as a black hole.

Quick start

Two quick start options are available for *intercept*:

- Download the latest intercept release.
- Clone the repo: `git clone git://github.com/session-replay-tools/intercept.git` .

Two quick start options are available for *tcpcopy*:

- Download the latest tcpcopy release.
- Clone the repo: `git clone git://github.com/session-replay-tools/tcpcopy.git` .

Getting intercept installed on the assistant server

1. cd intercept
2. ./configure
   - choose appropriate configure options if needed
3. make
4. make install

Configure Options for intercept

```
--single             run intercept at non-distributed mode
--with-pfring=PATH   set path to PF_RING library sources
--with-debug         compile intercept with debug support (saved in a log file)
```

Getting tcpcopy installed on the online server

1. cd tcpcopy
2. ./configure
   - choose appropriate configure options if needed
3. make
4. make install

Configure Options for tcpcopy

```
--offline                    replay TCP streams from the pcap file
--pcap-capture               capture packets at the data link
--pcap-send                  send packets at the data link layer instead of the IP 
layer
--with-pfring=PATH           set path to PF_RING library sources
--set-protocol-module=PATH  set tcpcopy to work for an external protocol module
--single                     if intercept and tcpcopy are both configured with "--
single" option,
                             only one tcpcopy works together with intercept,
                             and better performance is achieved.
--with-tcmalloc              use tcmalloc instead of malloc
--with-debug                 compile tcpcopy with debug support (saved in a log 
file)
```

Running TCPCopy

Assume *tcpcopy* and *intercept* are both configured with "./configure".

1) On the target server which runs server applications:

```
  Set route commands appropriately to route response packets to the assistant 
server

  For example:

     Assume 61.135.233.161 is the IP address of the assistant server. We set the
     following route command to route all responses to the 62.135.200.x's clients
     to the assistant server.

       route add -net 62.135.200.0 netmask 255.255.255.0 gw 61.135.233.161
```

2) On the assistant server which runs intercept(root privilege or the CAP_NET_RAW capability is required):

```
   ./intercept -F <filter> -i <device,>

   Note that the filter format is the same as the pcap filter.
   For example:

      ./intercept -i eth0 -F 'tcp and src port 8080' -d

      intercept will capture response packets of the TCP based application which
```

```
listens
      on port 8080 from device eth0
```

3) On the online source server (root privilege or the CAP_NET_RAW capability is required):

```
  ./tcpcopy -x localServerPort-targetServerIP:targetServerPort -s <intercept
server,>
  [-c <ip range,>]

  For example(assume 61.135.233.160 is the IP address of the target server):

    ./tcpcopy -x 80-61.135.233.160:8080 -s 61.135.233.161 -c 62.135.200.x

    tcpcopy would capture port '80' packets on current server, change client IP
address
    to one of 62.135.200.x series, send these packets to the target port '8080' of
the
    target server '61.135.233.160', and connect 61.135.233.161 for asking
intercept to
    pass response packets to it.

    Although "-c" parameter is optional, it is set here in order to simplify route
    commands.
```

Note

1. It is tested on Linux only (kernal 2.6 or above)
2. TCPCopy may lose packets hence lose requests
3. Root privilege or the CAP_NET_RAW capability(e.g. setcap CAP_NET_RAW=ep tcpcopy) is required
4. TCPCopy only supports client-initiated connections now
5. TCPCopy does not support replay for server applications which use SSL/TLS
6. For MySQL session replay, please refer to https://github.com/session-replay-tools
7. ip_forward should not be set on the assistant server
8. Please execute "./tcpcopy -h" or "./intercept -h" for more details.

Influential Factors

There are several factors that could influence TCPCopy, which will be introduced in detail in the following sections.

1. Capture Interface

tcpcopy utilizes raw socket input interface by default to capture packets at the network layer on the online server. The system kernel may lose some packets when the system is busy.

If you configure *tcpcopy* with "--pcap-capture", then *tcpcopy* could capture packets at the data link layer and could also filter packets in the kernel. With PF_RING, *tcpcopy* would lose less packets when using pcap capturing.

Maybe the best way to capture requests is to mirror ingress packets by switch and then divide the huge traffic to several machines by load balancer.

2. Sending Interface

tcpcopy utilizes raw socket output interface by default to send packets at the network layer to a target server. If you want to avoid ip_conntrack problems or get better performance, configure *tcpcopy* with "--pcap-send", then with appropriate parameters *tcpcopy* could send packets at the data link layer to a target server.

3.On the Way to the Target Server

When a packet is sent by *tcpcopy*, it may encounter many challenges before reaching the target server. As the source IP address in the packet is still the end-user's IP address(by default) other than the online server's, some security devices may take it for an invalid or forged packet and drop it. In this case, when you use tcpdump to capture packets on the target server, no packets from the expected end-users will be captured. To know whether you are under such circumstances, you can choose a target server in the same network segment to do a test. If packets could be sent to the target server successfully in the same network segment but unsuccessfully across network segments, your packets may be dropped halfway.

To solve this problem, we suggest deploying *tcpcopy*, *target applications* and *intercept* on servers in the same network segment. There's also another solution with the help of a proxy in the same network segment. *tcpcopy* could send packets to the proxy and then the proxy would send the corresponding requests to the target server in another network segment.

Note that deploying the target server's application on one virtual machine in the same segment may face the above problems.

4. OS of the Target Server

The target server may set rpfilter, which would check whether the source IP address in the packet is forged. If yes, the packet will be dropped at the network layer.

If the target server could not receive any requests although packets can be captured by tcpdump on the target server, you should check if you have any corresponding rpfilter settings. If set, you have to remove the related settings to let the packets pass through the network layer.

There are also other reasons that cause *tcpcopy* not working, such as iptables setting problems.

5. Applications on the Target Server

It is likely that the application on the target server could not process all the requests in time. On the one hand, bugs in the application may make the request not be responded for a long time. On the other hand, some protocols above TCP layer may only process the first request in the socket buffer and leave the remaining requests in the socket buffer unprocessed.

6. OS of the assistant Server

You should not set ip_forward true or the assistant server can't act as a black hole.

Release History

- 2014.09 v1.0 TCPCopy released

Bugs and feature requests

Have a bug or a feature request? Please open a new issue. Before opening any issue, please search for existing issues.

Technical support

Copyright and license

Releases 3

Fix slide window related problems and add option for tcmalloc Latest
on 18 Apr 2018

+ 2 releases

Packages

No packages published

Contributors 11

Languages

● C 99.3% ● C++ 0.7%