android系统 使用iptables 设置网络访问规则,(设置网址访问黑名单,白名单)

作者:wzb<wangzhibin_x@foxmail.com>

1.NetworkManagementService.java
系统源码中本身有相关的防火墙设置,通过iptables实现

```
1877
1878      @Override
1879      public void setFirewallEnabled(boolean enabled) {
1880          enforceSystemUid();
1881          try {
1882              mConnector.execute("firewall", enabled ? "enable" : "disable");
1883              mFirewallEnabled = enabled;
1884          } catch (NativeDaemonConnectorException e) {
1885              throw e.rethrowAsParcelableException();
1886          }
1887      }
1888
```

调用setFirewallEnabled() 时enable防火墙,会阻止所有网络访问

调用方法:
在具有系统权限的地方
import android.os.INetworkManagementService;
final INetworkManagementService netManager = INetworkManagementService.Stub
.asInterface(ServiceManager.getService(Context.NETWORKMANAGEMENT_SERVICE));
netManager.setFirewallEnabled(true); 即可

有些网络配置不能放在main 线程,需开个子线程
Runnable runnable=new Runnable() {

```
    @Override
    public void run() {
      // TODO Auto-generated method stub
          try {
              netManager.enableUrl("www.baidu.com");
              netManager.enableUrl("www.huayinghealth.com");
              //netManager.setFirewallEgressSourceRule("120.76.47.120",false);
              Log.d("wzb","set youku.com true");
           } catch (RemoteException e) {
              // ignored
              Log.d("wzb","set youku.com err");
           }
    }
  };
```

2.iptables具体实现地方
alps/system/netd/
FirewallController.cpp

```
int FirewallController::enableFirewall(void) {
    int res = 0;
    //if(1) return res;
    // flush any existing rules
    disableFirewall();
    // create default rule to drop all traffic
    res |= execIptables(V4, "-A", LOCAL_INPUT, "-j", "DROP", NULL);
    //res |= execIptables(V4V6, "-A", LOCAL_OUTPUT, "-j", "REJECT", NULL);
    res |= execIptables(V4, "-A", LOCAL_FORWARD, "-j", "REJECT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-A", LOCAL_MANGLE_POSTROUTING, "-j", "DROP", NULL);

    return res;
}

int FirewallController::disableFirewall(void) {
    int res = 0;

    // flush any existing rules
    res |= execIptables(V4V6, "-F", LOCAL_INPUT, NULL);
    res |= execIptables(V4V6, "-F", LOCAL_OUTPUT, NULL);
    res |= execIptables(V4V6, "-F", LOCAL_FORWARD, NULL);
    res |= execIptables(V4V6, "-t", "mangle", "-F", LOCAL_MANGLE_POSTROUTING, NULL);

    return res;
```

仿照系统的enablefirewall函数
写一个enbale dns ,和enable url函数

```
//add by wzb for test 20160623
int FirewallController::enableDNSPort(int protocol,int port) {
    char protocolStr[16];
    sprintf(protocolStr, "%d", protocol);

    char portStr[16];
    sprintf(portStr, "%d", port);

    int res = 0;

    res |= execIptables(V4, "-I", LOCAL_INPUT, "-p", protocolStr, "--sport", portStr, "-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-I", LOCAL_OUTPUT, "-p", protocolStr, "--dport", portStr, "-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING,"-p", protocolStr, "--sport", portStr, "-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING,"-p", protocolStr, "--dport", portStr, "-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING,"-p", protocolStr, "--sport", "80", "-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING,"-p", protocolStr, "--dport", "80", "-j", "ACCEPT", NULL);

    return res;
}

int FirewallController::enableUrl(const char* addr) {
    ALOGD("wzb22  addr=%s \n",addr);
    int res = 0;
    //if(1) return res;
    res |= execIptables(V4, "-I", LOCAL_INPUT, "-s", addr, "-j", "ACCEPT", NULL);
     res |= execIptables(V4, "-I", LOCAL_FORWARD, "-s", addr,"-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-I", LOCAL_OUTPUT, "-d", addr, "-j", "ACCEPT", NULL);
     res |= execIptables(V4, "-I", LOCAL_FORWARD, "-d", addr,"-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING, "-s", addr,"-j", "ACCEPT", NULL);
    res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING, "-d", addr,"-j", "ACCEPT", NULL);
    //res |= execIptables(V4, "-I", LOCAL_INPUT, "-s", "120.76.47.120", "-j", "ACCEPT", NULL);
    //res |= execIptables(V4, "-I", LOCAL_OUTPUT, "-d", "120.76.47.120", "-j", "ACCEPT", NULL);
    //res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING, "-s", "120.76.47.120","-j", "ACCEPT", NULL);
    //res |= execIptables(V4, "-t", "mangle", "-I", LOCAL_MANGLE_POSTROUTING, "-d", "120.76.47.120","-j", "ACCEPT", NULL);
    return res;
}
//end
```

注意:需要enable dns的port 53, web的port 80

3.黑名单实现原理:
直接将域名ip地址加入iptables drop规则即可

4.白名单实现原理:
先setenablefirewall,将所有网络禁掉

然后enbale dns port53,打开dns解析功能
enable web的port 80

将白名单域名的dns解析后的ip地址加入iptables accept规则.这样就可以实现白名单

```
//add by wzb for test
@Override
public void enableUrl(String url) {
    Slog.d("wzb", "enable url is: " + url);
    Preconditions.checkState(mFirewallEnabled);
    if(mFirewallEnabled) {
        try {
            mConnector.execute("firewall", "enable_dns_port", 53);
            Slog.d("wzb", "Firewall is enabled. Open dns port is: " + 53);
        } catch (NativeDaemonConnectorException e) {
            throw e.rethrowAsParcelableException();
        }
    }
    InetAddress[] ipArray;
    try {
        ipArray = InetAddress.getAllByName(url);
          if(ipArray != null) {
            for(int i = 0; i < 1/*ipArray.length*/; i++) {
                Slog.d("wzb", "ipArray[" + i +"].getHostAddress() = " + ipArray[i].getHostAddress());
                mConnector.execute("firewall", "enable_url", ipArray[i].getHostAddress());
            }

        }
    } catch (UnknownHostException e) {
        Slog.e("wzb", "Gets all IP addresses associated with the given host \"" + url +"\" faild. Because the host name can not be reso
        e.printStackTrace();
    } catch (NativeDaemonConnectorException e) {
        e.rethrowAsParcelableException();
    }
}
//end--
```

5.mConnector.execute() 是在system/netd/server/CommandListener.cpp里面解析的

```
int CommandListener::FirewallCmd::runCommand(SocketClient *cli, int argc,
        char **argv) {
    if (argc < 2) {
        cli->sendMsg(ResponseCode::CommandSyntaxError, "Missing command", false);
        return 0;
    }

    if (!strcmp(argv[1], "enable")) {
        int res = sFirewallCtrl->enableFirewall();
        return sendGenericOkFail(cli, res);
```

```
  //add by wzb for test 20160623
  if(!strcmp(argv[1], "enable_dns_port")) {
      // 3¶ÔÓ¦·¢ËÍ¶ËµÄ²ÎÊý¸öÊý
      if (argc < 3) {
              cli->sendMsg(ResponseCode::CommandSyntaxError, "Missing argument", false);
              return 0;
      }
      // portÎª·¢ËÍ¶ËµÄÄúÈý¸ö²ÎÊý
      int port = atoi(argv[2]);
      int res = 0;
      res |= sFirewallCtrl->enableDNSPort(PROTOCOL_UDP, port);
      res |= sFirewallCtrl->enableDNSPort(PROTOCOL_TCP, port);
      return sendGenericOkFail(cli, res);
  }

   if(!strcmp(argv[1], "enable_url")) {
      if (argc < 3) {
          cli->sendMsg(ResponseCode::CommandSyntaxError,
                                              "Missing argument",
                                              false);
                                  return 0;
      }
      const char* addr = argv[2];
      int res = sFirewallCtrl->enableUrl(addr);
      //int res = sFirewallCtrl->setEgressSourceRule(addr);

      return sendGenericOkFail(cli, res);
  }
  //end
```

实现enable dns 和url