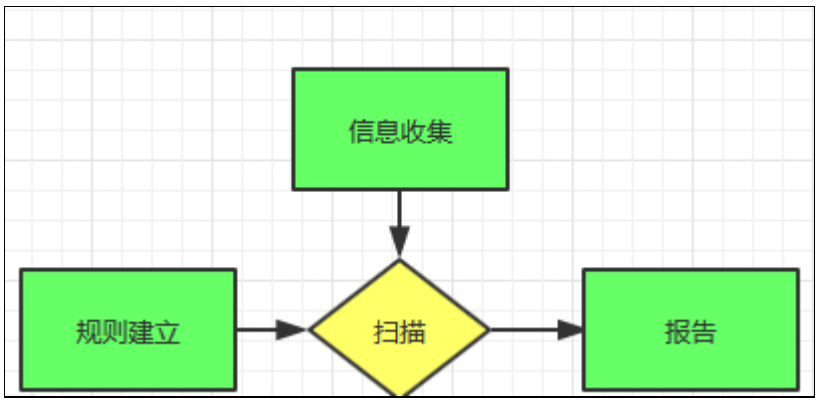
# 企业自研扫描器之路：信息收集模块

## 1、前言

随着甲方的安全工作进入了轨道，要提高工作效率以及减少重复工作的次数，迸发出了弄个工具的想法，主要是想弄个扫描器，而现有的 wvs、appscan 等这类工具，由于种种原因和环境，并不能满足我们的需求，所以只能走入自研之路。

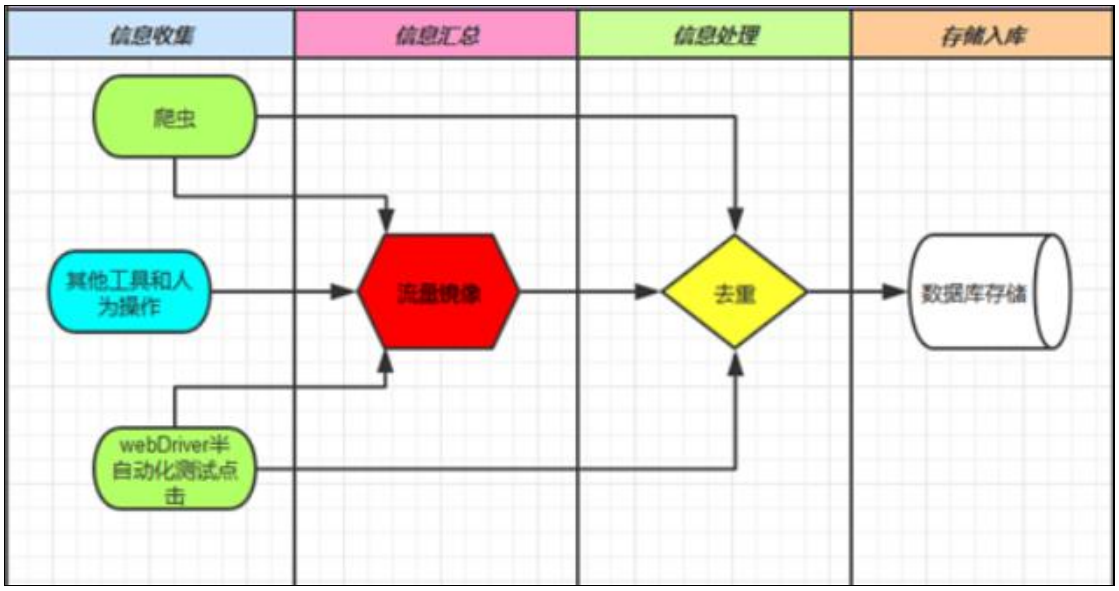
然后对于一个没有做过开发的人来说，体验开发的过程无疑是痛苦的，而对于自己本身又是需求者和设计者，无疑更痛苦。但经过与朋友的沟通和参考一些巨头公司同行流出的思路，于是拆分成下面的几个大的部分：




如今我们一直在信息收集模块奋斗中，不过我们也终于是见到了曙光，于是就先分享一下信息收集模块的思路和实现的步骤。

## 2、信息收集模块的总体设计

对于信息收集模块的设计，如下：

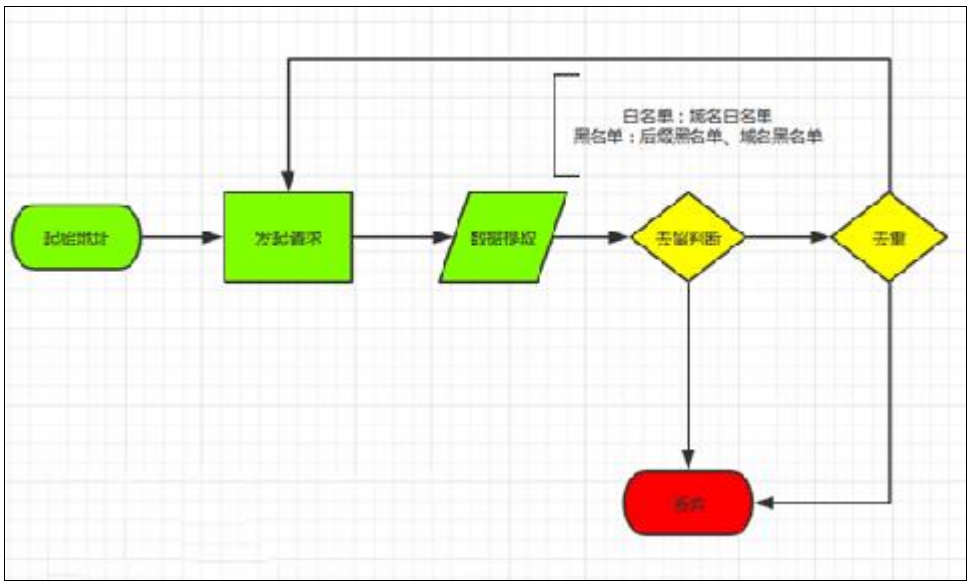



其实刚开始只考虑到爬虫，但是后期的交流才发现自己的落伍，原来流量镜像才是王道，使用流量镜像进行收集的话，好处很多，除了收集信息之外，其实还有许多好处，例如可以负责监控有没有新的产品和应用的上线等等。

不过流量镜像的收集有些被动，所以也就继续加入了爬虫模块，之后有朋友说他们公司的测试用了 webdriver，于是又研究了一些 webdriver，发现 webdriver 的一个特点实现模拟点击，也就解决了我对爬虫如何取 ajax 的请求值的问题了（其实爬虫也可以实现 ajax 的取值，例如开源的 Heritrix 中就有）。

## 3、爬虫的实现

爬虫的初步实现，比较容易，思路就是不断遍历新的地址，只不过在这个过程中需要考虑边界的问题，和防止重复爬去的问题，于是在设计的时候就加入了白名单和黑名单的设计：




首先是请求包的实现，这个直接采用了 httpclient.jar 来实现，可以直接实现 http 和 https 的请求，只需要考虑是 get 还是 post 方式，获取其实请求方式，可以加入 method 字段进行判断：

```
public HttpResponse sendReq (){
    if(httpRequest.getMethod().toLowerCase().equals("post")){
        post();
    }else{
        get();
    }
    return httpResponse;

}
```

之后是数据的提取，很多地址字段中的可能是相对路径，因此需要在处理的时候就加入域名和请求协议，在这个过程中，也可以进行黑白名单的检测，例如后缀的检测，以从 href 和 src 中提取为例：

仅供参考

```
//     提取 href/src 等属性

    public ArrayList dealHtml(String result){

        ArrayList urls_list = new ArrayList();
```

```
            String regex="<\\w.*?>";
            Pattern pattern = Pattern.compile(regex);
            Matcher matcher = pattern.matcher(result);
            StringBuffer content=new StringBuffer();
            while(matcher.find()){
                String temp=matcher.group();
   //             System.out.println(temp);
                String regex_attr="((\\w+)=([\"](\\S*)?[\"]
|[']([\\S*)?[']))>?";
   //             String regex_attr = "((\\w+)=[\"?](\\S*)
[\"?])";
                Pattern pattern_attr = Pattern.compile(rege
x_attr);
                Matcher matcher_attr = pattern_attr.matcher
(temp);
                while(matcher_attr.find()){
   //                System.out.println(matcher_attr.group
());
                    if(check_attr(matcher_attr.group(2).toL
owerCase())){
                        String temp1=matcher_attr.group(4);
                        if(temp1==null){
                            temp1=matcher_attr.group(5);
                        }
                        temp1=temp1.replaceAll("&amp;", "&
");
                        if(check_type(temp1)&&check_exclude
(temp1)){
   //                       System.out.println(matcher_attr.g
roup(3));
                            if(check_domain(temp1))
                                urls_list.add(temp1);
```

```
                            if(!check_url(temp1)){
                                if(temp1.startsWith("/"))
                                    urls_list.add("http://"
+DefaultConf.url_domain+temp1);
                                else
                                    urls_list.add("http://"
+DefaultConf.url_domain+"/"+temp1);
                            }
                        }
                    }
                }
            }
            return urls_list;

        }

        public boolean check_attr(String attr){
            String[] array_attrs = DefaultConf.array_attr;
            for(int i=0;i<array_attrs.length;i++){
                if(attr.toLowerCase().equals(array_attrs
[i])){

                    return true;
                }
            }
            return false;
        }

        public boolean check_domain(String url){
```

```
            String regex="https?://(\\S*"+DefaultConf.doma
in+")";
            Pattern pattern = Pattern.compile(regex);
            Matcher matcher = pattern.matcher(url);
            if(matcher.find()){
                return true;
            }else{

                return false;
            }
        }

        public boolean check_url(String url){
            if(url.startsWith("http://")||url.startsWith("
https://")){
                return true;
            }else{
                return false;
            }
        }

        public boolean check_type(String url){
    //        System.out.println(url);
            if(url.contains("?"))
                url=url.substring(0, url.indexOf("?"));
            String[] types = DefaultConf.types;
            for(int i=0;i<types.length;i++){
    //            System.out.println(url);
```

```
                if(url.substring(url.lastIndexOf(".")+1).t
oLowerCase().equals(types[i])){


                    return false;

                }

            }

            return true;

        }


        public boolean check_exclude(String url){

            String[] excludes = DefaultConf.exclude_url;

            for(int i=0;i<excludes.length;i++){

    //            System.out.println(url);

                if(url.toLowerCase().contains(excludes
[i])){


                    return false;

                }

            }

            return true;

        }
```

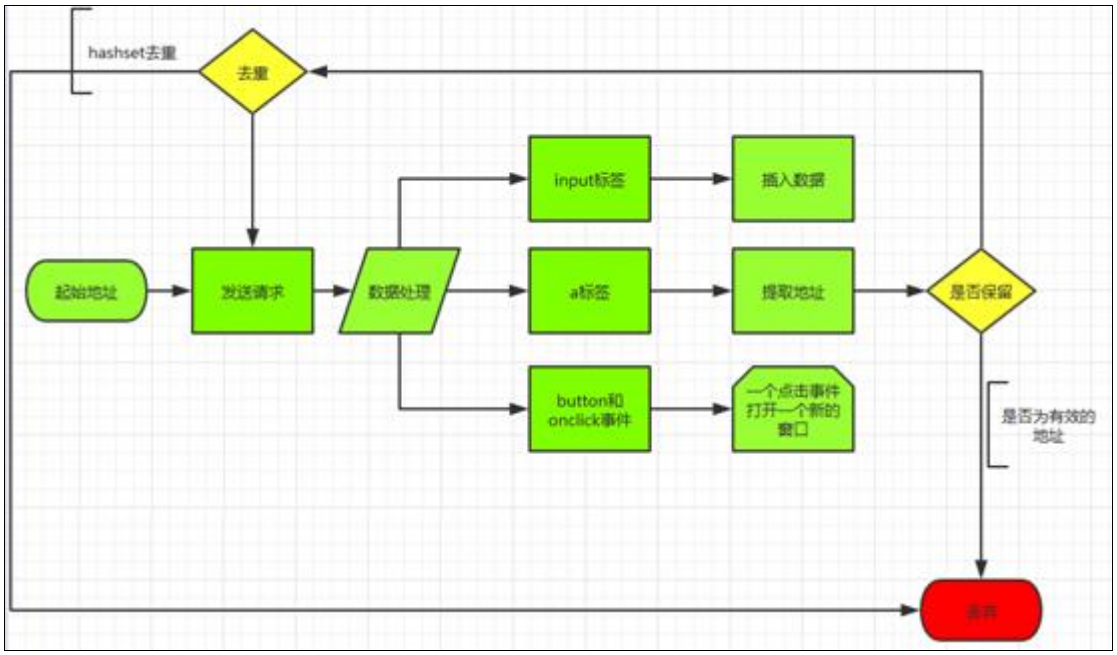
而去重的实现，这里直接就是用 hashset 进行去重

```
DealHtml dealHtml = new DealHtml();
ArrayList lists = new ArrayList();
lists=dealHtml.dealHtml(sb.toString());

for(int i=0;i<lists.size();i++){

    if(set.add(lists.get(i).toString().trim().toLowerCase())){
        System.out.println(lists.get(i));
        try {
        File file = new File(DefaultConf.outFileName);
        OutputStream out = new FileOutputStream(file,true);
        out.write((lists.get(i).toString()+"\r\n").getBytes());
        out.flush();

            out.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            logger.error(e);
        }
        httpRequest.setUrl(lists.get(i).toString());
        test(httpRequest);

       }
 }
```

至于实际的处理，入库前还有一步去重的检测.

## 4、WebDriver 的实现

WebDriver 的自动化测试，并不像我开始想的那么美好，开始以为直接给一个地址 ,他就开始自动的遍历 ,自动的点击 ,然而实际的操作并不如我想的那样，每一个页面的元素都有一个位置，你需要告诉 webdriver 位置在哪，以及对于这个位置的元素应该做什么操作，这就需要使用 XPath 来实现了，而 webDriver 在查找元素位置的时候，如果页面刷新了，就算是前进一步然后回退，之前定位的元素后所生成的 WebElement 值也会改变，就需要重新去匹配了，所以这对我之前的遍历造成了很大的困难 ,因为当我访问一个地址的时候 ,比如需要点击操作，而如果点击的是超链接，而这个超链接是本地标签打开的话，那我想继续操作下一个元素，就会出现异常，为了解决这个问题，我想了一个笨办法，就是一旦需要打开或者点击操作，我就新建一个标签，在重新打开页面，在新的页面上进行操作，操作完成后就关闭此页面，这样就不影响我对当前页面的操作了。




目前我实现的 webdriver 的提取元素的规则，主要针对 a、input、button 以及 onclick 事件，规则如下：

```
    ##提取规则

    #a 标签规则

    a=//a

    #input 标签规则

    input=//input[not(@value!='') and not(@type='hidden') and not(@type='checkbox') and not(@type='radio') and not(@disabled='disabled')]|//textarea

    #提交按钮规则

    sub=//*[@type='submit' or @type='button']|//button

    #onclick 事件点击

    on_click=//*[@onclick and not(@href) and not(contains(@onclick,'this'))]|//*[@onclick and not(contains(@href,'http')) and not(contains(@onclick,'this'))]
```

而提取后的操作，也需要自己设定，比如 input 标签的操作：

```
public void inputValue(WebDriver wd){
    System.out.println("----------------进入input模块----------------");
    List<WebElement> lists = (List<WebElement>) wd.findElements(By.xpath(Config.getInput()));
    for(WebElement link : lists){
        try{
        System.out.println(link.getAttribute("outerHTML"));
            if(Check.checkTab(link.getAttribute("outerHTML").toLowerCase())){
                link.clear();
                link.sendKeys("test123--");
                if(link.getAttribute("outerHTML").contains("身份证")){
                    System.out.println("input输入：身份证");
                    link.clear();
                    link.sendKeys("320[redacted]246");
                }
                if(link.getAttribute("outerHTML").contains("手机")||link.getAttribute("outerHTML").contains("电话")){
                    System.out.println("input输入：电话");
                    link.clear();
                    link.sendKeys("13800000000");
                }
```

## 5、流量镜像的实现

流量镜像的数据的提取，之前一直没有思路，好在同事 Ripz 提供了一个 httpcap 工具，直接就实现了对 cap 数据库中 http 数据的提取，之后其实就是对有效数据的提取了，我设计的时存储的几个字段：方法、url、host、还有 post 数据的 data，所以就是如何从解析后的 HTTP 数据包中提取有效的字段，并且规律存储，为此我们稍微修改了一下 httpcap 工具包，使得他输出的格式能够满足我们的需要，并且屏蔽了 response 数据的输出，同时我们要设计了白名单和黑名单，以下是提取字段的规则实现：

```
public Request getData(String data){
    Request req = new Request();
    String reg="-- -- -->.*\\[(\\d+\\.\\d+\\.\\d+\\.\\d+)?:.*\\].*(POST|GET)\\s(.*)\\s?HTTP.*Host:\\s*(\\S*)(.*)";
    Pattern pattern=Pattern.compile(reg);
    Matcher matcher =pattern.matcher(data);
    while(matcher.find()){
        System.out.println(matcher.group());
        req.setDesIP(matcher.group(1));
        req.setMethod(matcher.group(2));
        req.setUrl(matcher.group(3));
        req.setHost(matcher.group(4));
        if(matcher.group(2).equals("POST")){
            System.out.println(matcher.group(4));
            String temp=matcher.group(5);
            String temp_reg="---------->>\\t(.*)\\t<<----------";
            Pattern pattern_temp=Pattern.compile(temp_reg);
```

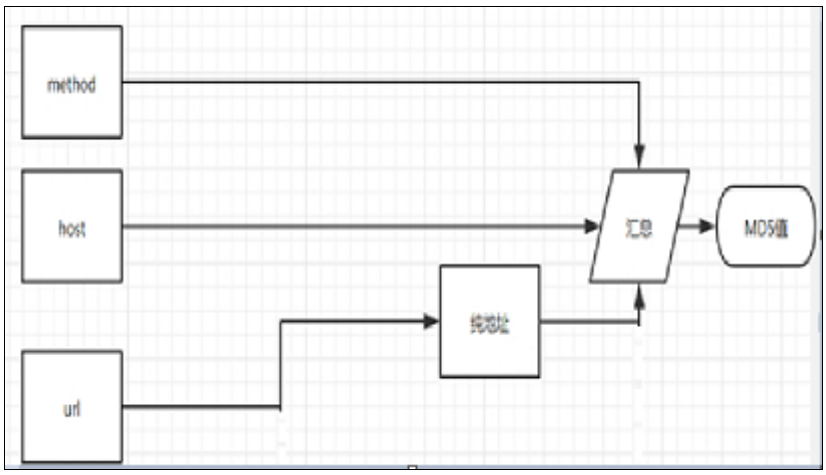
因此流量中提取数据也由最困难的部分成为了最简单的部分了。

## 6、数据处理的实现

经过上面的三种方法对地址的收集，之后的就是入库的操作，其实入库的操作很简单，问题是如果筛选数据，并且真正的去重，上面的数据只是简单的去重，但是并没有达到真正可以入库的标准，比较之前的 hashset 去重，考虑的是全部

匹配后去重，例如参数的问题，id=1 和 id=2 的 hash 结果就不一样，但是对于我们来说，id 就是一个参数，应该就记录一次就行了。

因此针对我考虑的问题，我决定生成 md5 值，进行比较，而针对哪些数据进行 md5 值的生成，我们进行了规划：




具体的实现，仅供参考：

```
public String getMd5(Request req){
    try{
    Comparator comparator = Collator.getInstance(java.util.Locale.ENGLISH);
    String url=req.getMethod()+req.getHost();
    String temp_url=req.getUrl();
    String temp_data=req.getData();
    if(temp_url.contains("?")){
        url=url+temp_url.substring(0, temp_url.indexOf("?")+1);
        temp_url=temp_url.substring(temp_url.indexOf("?")+1);
        String[] a=temp_url.split("&");
        Arrays.sort(a,comparator);
        for(String b:a){
            if(b.contains("=")){
                String[] c=b.split("=");
                url=url+c[0];
            }else{
                url=url+b;
            }
        }

    }else{
        url=url+temp_url;
    }
    try{
    if(!temp_data.equals(null)&&!temp_data.equals("")){
        if(temp_data.contains("&")){
            String[] a=temp_url.split("&");
            Arrays.sort(a,comparator);
            for(String b:a){
                if(b.contains("=")){
                    String[] c=b.split("=");
                    url=url+c[0];
                }else{
                    url=url+b;
                }
            }

        }else{
            if(temp_data.contains("=")){
                String[] c=temp_data.split("=");
                url=url+c[0];
```