安全组件参考说明

- java安全编码规范
- JAVA安全SDK，SDK介绍详见下述。

目录

1、引用MScurity Lib包

环境需求

- Java 6及以上（包含6）
- Maven 3

a、编译jar包：

```
    mvn -Dmaven.test.skip=true clean install
```

b、加入本地maven仓库（如果有公司内部maven仓库，传入maven仓库更方便）：

```
    mvn install:install-file -Dfile=${project.basedir}/target/security-2.1-
release-jar-with-dependencies.jar -DgroupId=com.corp.security -
DartifactId=security -Dversion=2.1-release -Dpackaging=jar
```

c、pom文件引入MScurity Lib包:

需要在自己的maven工程pom.xml中加入如下依赖:

```
    <dependency>
        <groupId>com.corp.security</groupId>
        <artifactId>security</artifactId>
        <version>2.1-release</version>
    </dependency>
```

2、SQL注入防护

a、引入SecurityUtil:

```
    import com.corp.vul.SecurityUtil;
```

b、针对字符串型转义:

字符串型参数需要用单引号包裹，默认不加单引号

```
    String id = "1' or '1'='1' #";
    String idEncode = SecurityUtil.JdbcSecuritySQLI(id);
    String query = "SELECT NAME FROM users WHERE id = '" + idEncode + "'";

    String idEncode = SecurityUtil.JdbcSecuritySQLI(id, true);
    String query = "SELECT NAME FROM users WHERE id = " + idEncode; //如果传入
true，则无需手动增加单引号
```

转义前后对比:

```
    转义前：SELECT NAME FROM users WHERE id = '1' or '1'='1' #'
    转义后：SELECT NAME FROM users WHERE id = '1\' or \'1\'\=\'1\' \#'
```

c、针对数字型转义:

```
    String sqli_n="1 union select * from users";
    String rs4 = SecurityUtil.SecuritySQLI(sqli_n);
    String query = "SELECT * FROM users WHERE id = " + rs4;
```

转义前后对比:

```
    转义前：SELECT * FROM users WHERE id = 1 union select * from users
    转义后：SELECT * FROM users WHERE id = 1 &#117;nion &#115;elect * from admin
```

3、xss防护

a、引入SecurityUtil:

```
    import com.corp.vul.SecurityUtil;
```

b、如果输出到html body:

```
    String rs = SecurityUtil.SecurityXssScript("<scritp>alert(1)</script>",
"h");
```

过滤前后对比:

```
    过滤前：<script> alert('xss') </script>
    过滤后：&lt;script&gt;alert&#x28;&#x27;xss&#x27;&#x29;&lt;&#x2f;script&gt;
```

c、如果输出到uri中:

```
    String rs = SecurityUtil.SecurityXssScript("javascript:alert(1);", "u");
```

过滤前后对比:

```
    过滤前：javascript:alert(1);
    过滤后：javascript%3Aalert%281%29%3B
```

d、如果输出到css中:

```
    String rs = SecurityUtil.SecurityXssScript("<div
style='xss:expression(alert(1));'></div>", "c");
```

过滤前后对比:

```
    过滤前：<div style='xss:expression(alert(1));'></div>
    过滤后：\3C div style=\22 xss:expression(alert(1));\22 \3E \3C \2F div\3E
```

e、如果输出到JavaScript代码块中:

```
    String rs = SecurityUtil.SecurityXssScript("alert('xss');", "j");
```

过滤前后对比:

```
    过滤前：alert('xss');
    过滤后：alert\x28\x27xss\x27\x29\x3B
```

4、url重定向防护

a、引入SecurityUtil:

```
import com.corp.vul.SecurityUtil;
```

b、自定义白名单:

```
String white_url="*.moresec.cn";
```

白名单支持xxx.moresec.cn和*.moresec.cn两种方式

c、校验url:

```
try{
    String url1 = "test.moresec.cn";
    Boolean rs2 =  SecurityUtil.SecurityUrlRedirect(url1 , white_url);
} catch (Exception e) {
    ...
}
```

rs2为true则为校验通过，false为校验不通过

5、SSRF防护

a、引入SecurityUtil:

```
import com.corp.vul.SecurityUtil;
```

b、自定义白名单:

```
String white_url="*.moresec.cn";
```

白名单支持xxx.moresec.cn和*.moresec.cn两种方式

c、校验url:

```
try{
    String url1 = "test.moresec.cn";
    Boolean rs2 =  SecurityUtil.SecuritySSRF(url1 , white_url);
} catch (Exception e) {
    ...
}
```

rs2为true则为校验通过，false为校验不通过

6、xxe防护

a、引入SecurityUtil:

```
import com.corp.vul.SecurityUtil;
```

b、对传入xml内容进行校验:

```
String xxe_s=
"\t <?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n" +
"\t <!DOCTYPE foo [ \n" +
"\t <!ENTITY myentity SYSTEM \"file:///etc/passwd\" >]>\n" +
"\t <abc>&myentity;</abc>";
boolean rs7 = SecurityUtil.SecurityXXE(xxe_s);
```

rs7为true则为校验通过，false为校验不通过

7、CRLF注入防护

a、引入SecurityUtil:

```
import com.corp.vul.SecurityUtil;
```

b、对插入header头的内容进行过滤:

```
String bad_value = "%0a%0dlocation:www.moresec.cn";
String health_value = SecurityUtil.SecurityRSHeaderInjection(bad_value);
```

c、过滤前后对比:

```
过滤前：%0a%0dlocation:www.moresec.cn
过滤后：location:www.moresec.cn
```

8、文件上传防护

a、引入SecurityUtil:

```
import com.corp.vul.SecurityUtil;
```

b、对文件名进行白名单校验:

```
String bad_filename="test.jsp";
String white_extname="txt|doc|docx|pdf|xls|xlsx|jpg|png|cvs|ppt|pptx";
boolean rs9 = SecurityUtil.isValidFileName(bad_filename , white_extname);
```

rs9为true则为校验通过，false为校验不通过

9、CSRF防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、生成32位长度的csrf_token，并在服务端保存好

```
    String csrf_token = SecurityUtil.getRand();
```

c、前端页面上加入csrf_token

```
<input type="hidden" name="csrf_token" value="${(_csrf.token)!}">
```

d、当前端发送请求给后端时进行验证

```
    String session = csrf_token;
    boolean rs5 = SecurityUtil.isCSRFProtectPassed(session,csrf_token);
```

rs5为true则校验通过，false则校验不通过

10.Mongdb注入防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、对参数进行过滤转义

```
    String id = "admin\"});//";
    SecurityUtil.SecurityMongodb(id);
    BasicDBObject databaseQuery = new BasicDBObject("user", id);
```

过滤前后代码比较：

```
    转义前：db.run.find({"user":"admin"});//"})
    转义后：db.run.find({"user":"admin\";"})
```

11.命令注入防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、检查ip是否符合规则

```
    String ip = "192.168.1.25";
    boolean rs11 = SecurityUtil.SecuritycheckIp(ip);
```

ip符合规则时rs11返回true，否则返回false

c、白名单设置访问路径

```
String[] whitelist = {"admin","user","guest","test"};
String command="test && ipconfig";
String rsdir = SecurityUtil.SecurityCominject(command,whitelist);
Runtime runtime=Runtime.getRuntime();
Process process=runtime.exec(new String[]{"dir ", rsdir});
```

经过白名单检测后

```
检测前：dir test && ipconfig
检测后：dir test
```

12.代码注入防护

a、引入SecurityUtil：

```
import com.corp.vul.SecurityUtil;
```

b、通过白名单创建对象

```
Map whiteList = new HashMap();
whiteList.put("Test",Test.class);
whiteList.put("ABC",ABC.class);
SecurityUtil.CodeSecurityInject(whiteList,"Test");
```

输入其他类的时候将不做创建

13.任意文件遍历防护

a、引入引入SecurityUtil：

```
import com.corp.vul.SecurityUtil;
```

b、列出允许访问的白名单列表

```
List whitelist = new ArrayList();
//设置允许访问的目录
whitelist.add("image");
whitelist.add("page");
whitelist.add("css");
String directory = "image/test/../..%5c..%2f../test.txt";
```

c、通过函数过滤

```
SecurityUtil.SecurityDir(whitelist,directory)
```

最后将输出

```
image/test//5c2f/test.txt
```

14.Xpath注入防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、将需要查询内容参数化

```
    List<String> param = new ArrayList();
    String username = "abc' or '1'='1";
    String password = "abc' or '1'='1";
    param.add(username);
    param.add(password);
    String retrieval = "//user[username=$username and
password=$password]/id/text()";
    String[] Name = {"$username","$password"};
    XPathExpression expr =
SecurityUtil.SecurityXPathInject(param,Name,retrieval);
    Object result = expr.evaluate(doc, XPathConstants.NODESET);
```

方法原型：

```
SecurityXPathInject(
    List param,
    String[] Name,
    String retrieval
)
```

Name数组中放置以param列表为参数顺序的retrieval的变量名

方法使用前后对比

```
    使用前：//user[username='abc' or '1'='1' and password='abc' or
'1'='1']/id/text()  //返回全部数据
    使用后：     //将不返回任何数据
```

15.反序列化漏洞防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、通过安全方法设置允许反序列白名单

```
ObjectInputStream in = new SecurityUtil.SecureObjectInputStream(fileIn,list);
```

如下列程序所示

```
    String[] list = {"com.test.mainTest$test","com.test.mainTest$Employee"};
    FileInputStream fileIn = new FileInputStream("C:\\test\\employee.ser");
    ObjectInputStream in = new
SecurityUtil.SecureObjectInputStream(fileIn,list);
    e1 = (Employee) in.readObject();
```

list中的类名可以通过getClass()获得

如果要反序列化的对象不在list数组中存在，将会报ClassNotFoundException错误

```
    java.lang.ClassNotFoundException: com.test.mainTest$Employee not found
```

只需要捕获该异常

```
    catch(ClassNotFoundException c){
        System.out.println("Employee class not found");
        return;
    }
```

16.WebSocket劫持防护

a、引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

b、获取请求头的Origin和白名单比较

```
    String HeaderOrigin = request.getHeader("Origin");
    String origin = "https://www.test.com";
    Boolean rs16 = SecurityUtil.SecurityWebSocket(origin,HeaderOrigin);
```

rs16返回flase，将对比失败；返回true，将对比成功。

17.逻辑漏洞防护

- ## 整数溢出安全方法

  ### a、引入SecurityUtil：

  ```
      import com.corp.vul.SecurityUtil;
  ```

  ### b、使用安全方法进行加减乘

  ```
      int a = 2147483647;
      System.out.println(SecurityUtil.addExact(a,2));
      int b = -2147483647;
      System.out.println(SecurityUtil.subtractExact(b,2));
      int c = 1073741848;
      System.out.println(SecurityUtil.multiplyExact(c,2));
  ```

如果发生溢出，则方法返回一个ArithmeticException异常

```
Exception in thread "main" java.lang.ArithmeticException: integer overflow
```

支持int和long两种类型

- # 入参判断安全防护

a、引入SecurityUtil：

```
import com.corp.vul.SecurityUtil;
```

b、检查数字大小

```
int i = 0;
int request = 12;
SecurityUtil.checkMM(request,i,"min");//另一种type为max
```

```
返回：
    因12<0不成立
    报"java.lang.Exception: Comparative failure!"错误
```

c、设置取值范围

```
int request = -2;
boolean rs17 = SecurityUtil.checkRange(request,"1-600");
```

如果rs17变量返回true，则request值在取值范围内，否则返回false

d、类型判断

```
String request = "admin";
SecurityUtil.checkType(request,"user");
```

admin和user类型不同，返回java.lang.Exception: Type Error 错误

e、签名校验

```
String ASE_KEY="aabbccddeeffgghh";//AESKEY 为16bytes
String content = "abcd";
String encryptResult = SecurityRandom.encrypt(content,ASE_KEY);
String decryptResult = SecurityRandom.decrypt(encryptResult,ASE_KEY);
if(SecurityRandom.checkAutograph(decryptResult,content)){
    //Doing something
    System.out.println("True");
}else {
    //no do something
    System.out.println("False");
}
```

True为校验成功，否则校验失败

f、页面删除越权防护

```
        String PageId = "82";
        String UserId = "guest";
        //sql查询PageId语句
        //返回rs
        while (rs.next()) {
            // 通过字段检索
            String id = rs.getString("PageId");
            String name = rs.getString("UserId");
            if(PageId.equals(id)){
                //userid是当前页面请求时候的用户的id,
                //name是页面创建的时候创建人的id
                if(SecurityUtil.checkPower(userid, name)){
                    System.out.println("删除成功！");
                }else {
                    System.out.println("删除失败！");
                }
            }
        }
```

SecurityUtil.checkPower返回的值决定发起请求删除的用户是否为创建页面的用户

如果不是，则返回删除失败。

18.敏感信息防护

a、去除敏感信息

在前端等注释信息应及时去除(JS脚本、html中的注释、账号秘密、特殊链接)

b、通过安全函数处理加密敏感信息

引入SecurityUtil：

```
    import com.corp.vul.SecurityUtil;
```

1. 身份证

   身份证将只返回前五位地区数

   ```
       String sfz = "370205621219253";
       System.out.println(SecurityUtil.filterSFZ(sfz));
   ```

   ```
       返回:37020**********
   ```

2. 手机号

   手机号返回前4位

   ```
       String phone = "13900442200";
       System.out.println(SecurityUtil.filterTel(phone));
   ```

```
返回：1390**********
```

3. 姓名

   返回姓名第一个字符

```
String phone = "测试测试测试";
System.out.println(SecurityUtil.filterName(phone));
```

```
返回：测*****
```

4. IP

   ip只返回a段

```
String ip = "192.168.1.1";
System.out.println(SecurityUtil.filterIP(ip));
```

   返回192.xxx.xxx.xxx

c、特殊情况加密

特殊情况可以根据客户使用aes加密

引入SecurityRandom包

```
import com.corp.vul.SecurityRandom;
```

AES加密

```
String ASE_KEY="aabbccddeeffgghs";//AES 为16bytes
String content = "test";
String encryptResult = SecurityRandom.encrypt(content,ASE_KEY);
String decryptResult = SecurityRandom.decrypt(encryptResult,ASE_KEY);
```

encryptResult是最后的加密结果，decryptResult是最后的解密结果

测试代码如下：

```
import com.corp.vul.SecurityUtil;

public class TestCode {

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

        //xss
        String rs0 = SecurityUtil.SecurityXssScript("<div
style=\"xss:expression(alert(1));\"></div>", "c");
        String rs1 = SecurityUtil.SecurityXssScript("<div
style=\"xss:expression(alert(1));\"></div>", "h");
        String rs2 = SecurityUtil.SecurityXssScript("<div
style=\"xss:expression(alert(1));\"></div>", "j");
        String rs3 = SecurityUtil.SecurityXssScript("<div
style=\"xss:expression(alert(1));\"></div>", "u");
```

```
        //sqli
        String sqli_s="name-time' or '1'='1' #";
        String sqli_n="1 union select * from admin";
        String rs4 = SecurityUtil.SecuritySQLI(sqli_n);
        String rs8 = SecurityUtil.JdbcSecuritySQLI(sqli_s);

        //url redirect
        String white_url = "*.moresec.cn";
        String url1 = "test.moresec.cn";
        Boolean rs5 =  SecurityUtil.SecurityUrlRedirect(url1 , white_url);


        //ssrf
        Boolean rs6 =  SecurityUtil.SecuritySSRF(url1 , white_url);

        //xml
        String xxe_s=
                "\t <?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n" +
                        "\t <!DOCTYPE foo [ \n" +
                        "\t <!ENTITY myentity SYSTEM \"file:///etc/passwd\"
>]>\n" +
                        "\t <abc>&myentity;</abc>";
        boolean rs7 = SecurityUtil.SecurityXXE(xxe_s);


        //crlf
        String bad_value = "%0a%0dlocation:www.moresec.cn";
        String health_value = SecurityUtil.SecurityRSHeaderInjection(bad_value);

        //file upload
        String bad_filename="test.jsp";
        String white_extname="txt|doc|docx|pdf|xls|xlsx|jpg|png|cvs|ppt|pptx";
        boolean rs9 = SecurityUtil.isValidFileName(bad_filename ,
white_extname);

        //csrf
        String csrf_token = SecurityUtil.getRand();
        String session = csrf_token;
        boolean csrf = SecurityUtil.isCSRFProtectPassed(session,csrf_token);

        //mongodb
        String mog = "admin\"});//";
        String mogd = SecurityUtil.SecurityMongodb(mog);

        //command injection
        String ip = "192.168.1.25";
        boolean ipc = SecurityUtil.SecuritycheckIp(ip);

        //directory traversal
        List whitelist = new ArrayList();
        whitelist.add("image");
        whitelist.add("page");
        whitelist.add("css");
        String directory = "image/test/../..%5c..%2f../test.txt";
        System.out.println(SecurityUtil.SecurityDir(whitelist,directory));

        //Xpath注入
```

```
        List<String> param = new ArrayList();
        String username = "root";
        String password = "root";
        param.add(username);
        param.add(password);
        String retrieval = "//user[username=$username and
password=$password]/id/text()";
        String[] Name = {"$username","$password"};
        DocumentBuilderFactory factorys = DocumentBuilderFactory.newInstance();
        factorys.setNamespaceAware(true);
        DocumentBuilder builder = factorys.newDocumentBuilder();
        Document doc = builder.parse("books.xml");
        XPathExpression expr =
SecurityUtil.SecurityXPathInject(param,Name,retrieval);
        Object result = expr.evaluate(doc, XPathConstants.NODESET);
        NodeList nodes = (NodeList) result;
        for (int i = 0; i < nodes.getLength(); i++) {
            System.out.println("xpath id ="+nodes.item(i).getNodeValue());
        }

        //Java 反序列化防护
        Employee e1 = null;
        String[] list = {"com.test.mainTest$test","com.corp.vul.Employee"};
        try
        {
            FileInputStream fileIn = new FileInputStream("employee.ser");
            ObjectInputStream in = new
SecurityUtil.SecureObjectInputStream(fileIn,list);
            e1 = (Employee) in.readObject();
            in.close();
            fileIn.close();
        }catch(IOException i)
        {
            i.printStackTrace();
            return;
        }catch(ClassNotFoundException c)
        {
            c.printStackTrace();
        }
        System.out.println(e1.mailCheck());

        //WebSocket
        String HeaderOrigin="https://www.t.com";
        //String HeaderOrigin = request.getHeader("Origin");
        String origin = "https://www.test.com";
        Boolean webs = SecurityUtil.SecurityWebSocket(origin,HeaderOrigin);

        //integer overflow
        try {
            int a = 2147483647;
            System.out.println(SecurityUtil.addExact(a,2));
            int b = -2147483647;
            System.out.println(SecurityUtil.subtractExact(b,2));
            int c = 1073741848;
            System.out.println(SecurityUtil.multiplyExact(c,2));
        }catch (ArithmeticException e){
            System.err.println("integer overflow");
        }
```

```
//检测大小
int i = 0;
int request = 12;
try {
    SecurityUtil.checkMM(request,i,"min");
}catch (Exception e){
    System.err.println("Comparative failure!");
}

//检测取值范围
int request1 = -2;
boolean rs17 = SecurityUtil.checkRange(request1,"1-600");
System.out.println(rs17);

//判断类型
String request2 = "admin";
try {
    SecurityUtil.checkType(request2,"user");
}catch (Exception e){
    System.err.println("Type Error");
}

//签名校验
String ASE_KEY="aabbccddeeffgghh";//AESKEY 为16bytes
String content = "abcd";
String encryptResult = SecurityRandom.encrypt(content,ASE_KEY);
String decryptResult = SecurityRandom.decrypt(encryptResult,ASE_KEY);
if(SecurityRandom.checkAutograph(decryptResult,content)){
    //Doing something
    System.out.println("Signature True");
}else {
    //no do something
    System.out.println("Signature False");
}

String PageId = "82";
String userid = "guest";
//sql查询PageId语句
//返回rs
//while (rs.next()) {
    // 通过字段检索
    //String id = rs.getString("PageId");
    //String name = rs.getString("UserId");
String id = "82";
String name = "guest";
    //userid是当前页面请求时候的用户的id,
    //name是页面创建的时候创建人的id
    if(SecurityUtil.checkPower(userid, name)){
        System.out.println("删除成功！");
    }else {
        System.out.println("删除失败！");
    }

//}

//sfz
String sfz = "370205621219253";
```

```
        System.out.println(SecurityUtil.filterSFZ(sfz));

        //sjh
        String phone = "13900442200";
        System.out.println(SecurityUtil.filterTel(phone));

        //name
        String names = "测试测试测试";
        System.out.println(SecurityUtil.filterName(names));

        //IP
        String ips = "192.168.1.1";
        System.out.println(SecurityUtil.filterIP(ips));

        System.out.println(rs0);
        System.out.println(rs4);
        System.out.println(rs8);
        System.out.println(health_value);
        System.out.println(rs9);
        System.out.println(csrf);
        System.out.println(mogd);
        System.out.println(ipc);
        System.out.println(webs);
    }
}
```