

谈谈源码泄露 · WEB 安全

1. 漏洞成因

在WEB安全体系当中，可能你对SQL注入，XSS跨站一些漏洞已经耳熟能详了，而源码泄露问题对于大部分开发者来说就相对陌生了，而源码泄露导致的问题却并不少见，在过往的泄露案例当中，不仅是小网站有此问题，在一些大的厂商同样出现不少，并因此拿到webshell。

比如在一些小型企业，可能公司并没有专门的服务器，而是把网站部署在某一个虚拟主机上面，代码文件比较多时FTP上传是比较慢的，于是开发者把代码先打包压缩后再上传，上传成功后再去服务器解压，这有虽然解决了上传速度慢的问题，不过却留下了一些安全隐患。

压缩包解压后如果没有删除，当攻击者发现后就可以把代码压缩包下载；因为部署到服务器上的都是源代码，这个时候攻击者就可以通过代码进一步挖掘一些安全漏洞：文件上传，SQL注入等。

2. GIT源码泄露

2.1 漏洞成因

当在一个空目录执行 `git init` 时，Git 会创建一个 `.git` 目录。这个目录包含所有的 Git 存储和操作的对象。如果想备份或复制一个版本库，只需把这个目录拷贝至另一处就可以了。

该目录的结构如下所示：

HEAD

config*

description

hooks/

info/

objects/

refs/

在这些结构中description 文件仅供 GitWeb 程序使用，我们可以无需关心。

- config 文件包含项目特有的配置选项。
- info 目录包含一个全局性排除（global exclude）文件，用以放置不希望被记录在 .gitignore 文件中的忽略模式（ignored patterns）。
- hooks 目录包含客户端或服务端的钩子脚本（hook scripts）

而在剩下的四个条目很重要：

- HEAD 文件指示目前被检出的分支；
- index 文件保存暂存区信息；
- objects 目录存储所有数据内容；
- refs 目录存储指向数据（分支）的提交对象的指针；

而在发布代码的时候，如果没有把.git这个目录删除，直接发布到了运行目录中。攻击者就可以通过这个文件夹，可以用来恢复源代码。

<http://www.localhost.test/.git/config>

通常会用到的利用工具GitHack，这个工具下载下来之后操作也特别简单。

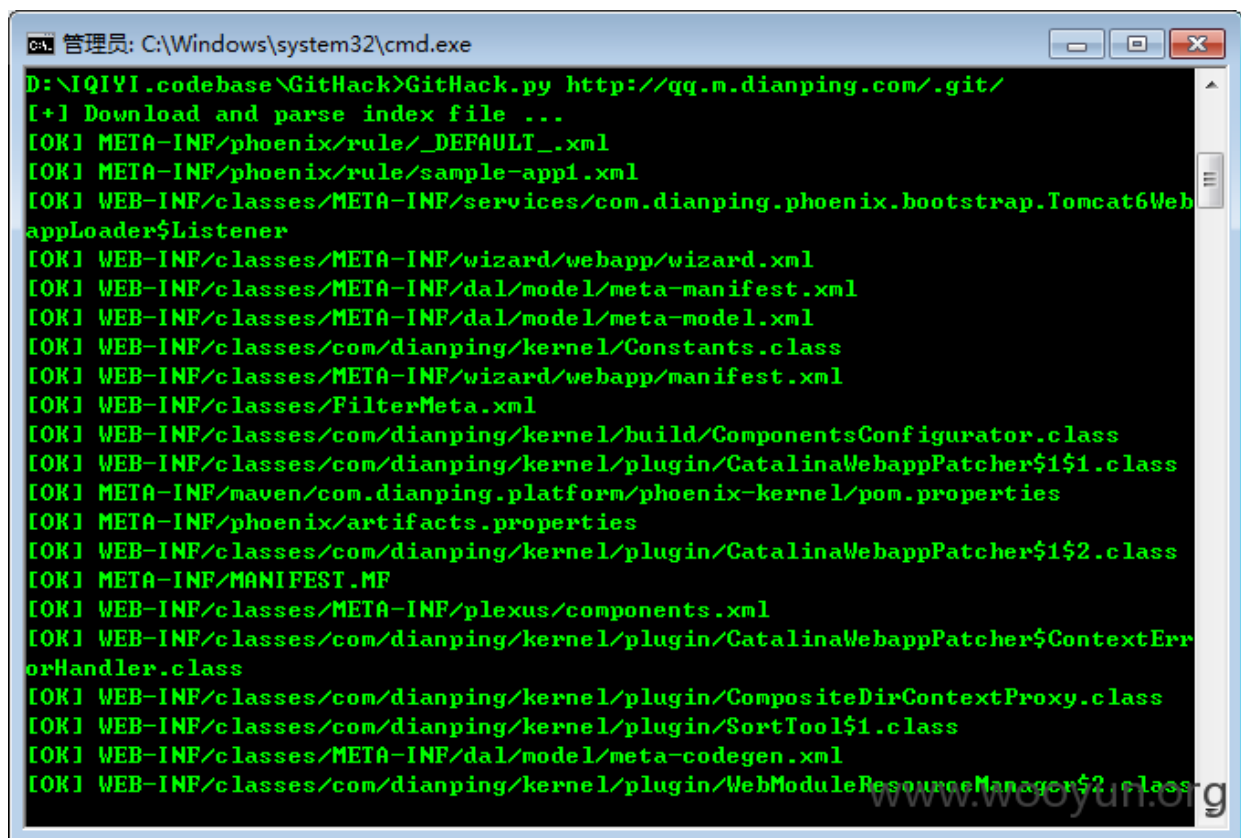
GitHack.py <http://www.localhost.test/.git/>

它能解析.git/index文件，并找到工程中所有的：文件名和文件sha1，然后去.git/objects/文件夹下下载对应的文件，通过zlib解压文件，按原始的目录结构写入源代码。

2.2 GIT 源码泄露 案例

2015年5月，乌云白帽子“lijiejie”提交漏洞“大众点评某站点git泄漏源代码”，缺陷编号：wooyun-2015-0117332

在此次案例当中，白帽子发现URL（<http://qq.m.dianping.com/.git/>）可以访问，于是通过工具githack下载里面的文件，下面为githack.py执行中的截图：



```
管理员: C:\Windows\system32\cmd.exe
D:\IQUIYI.codebase\GitHack>GitHack.py http://qq.m.dianping.com/.git/
[+] Download and parse index file ...
[OK] META-INF/phoenix/rule/_DEFAULT_.xml
[OK] META-INF/phoenix/rule/sample-app1.xml
[OK] WEB-INF/classes/META-INF/services/com.dianping.phoenix.bootstrap.TomcatWeb
appLoader$Listener
[OK] WEB-INF/classes/META-INF/wizard/webapp/wizard.xml
[OK] WEB-INF/classes/META-INF/dal/model/meta-manifest.xml
[OK] WEB-INF/classes/META-INF/dal/model/meta-model.xml
[OK] WEB-INF/classes/com/dianping/kernel/Constants.class
[OK] WEB-INF/classes/META-INF/wizard/webapp/manifest.xml
[OK] WEB-INF/classes/FilterMeta.xml
[OK] WEB-INF/classes/com/dianping/kernel/build/ComponentsConfigurator.class
[OK] WEB-INF/classes/com/dianping/kernel/plugin/CatalinaWebappPatcher$1$1.class
[OK] META-INF/maven/com.dianping.platform/phoenix-kernel/pom.properties
[OK] META-INF/phoenix/artifacts.properties
[OK] WEB-INF/classes/com/dianping/kernel/plugin/CatalinaWebappPatcher$1$2.class
[OK] META-INF/MANIFEST.MF
[OK] WEB-INF/classes/META-INF/plexus/components.xml
[OK] WEB-INF/classes/com/dianping/kernel/plugin/CatalinaWebappPatcher$ContextErr
orHandler.class
[OK] WEB-INF/classes/com/dianping/kernel/plugin/CompositeDirContextProxy.class
[OK] WEB-INF/classes/com/dianping/kernel/plugin/SortTool$1.class
[OK] WEB-INF/classes/META-INF/dal/model/meta-codegen.xml
[OK] WEB-INF/classes/com/dianping/kernel/plugin/WebModuleResourceManager$2.class
```

源码被下载下来之后，白帽子打开其中的一个代码文件可以看到里面的源码：



```
Java Decompiler
File Edit Navigate Help
Constants.class HealthCheckServlet.class x
package com.dianping.kernel.servlet;

import com.dianping.kernel.Constants;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.io.FileUtils;
import org.apache.log4j.Logger;

public class HealthCheckServlet extends HttpServlet
{
    private static final long serialVersionUID = 6182573529791601378L;
    private Logger m_logger;
    private static final String OP_OK = "ok";
    private static final String OP_FAILED = "failed";
    private static final String ONLINE_INFO = "online";
```

3. SVN导致文件泄露

3.1 漏洞成因

SVN是Subversion的简称，是一个开放源代码的版本控制系统，相较于RCS、CVS，它采用了分支管理系统，它的设计目标就是取代CVS。互联网上很多版本控制服务已从CVS迁移到Subversion。

很多网站都使用了svn版本控制系统，和使用git版本控制器类似，很多开发者网站安全意识不足，代码放到生产环境中后，没有清理svn的一些信息，导致svn残留，因此攻击者可以使用工具dvcs-ripper下载网站源码。

此工具的Github地址：<https://github.com/kost/dvcs-ripper>

利用命令如下：

```
rip-svn.pl -v -u http://www.localhost.test/.svn/
```

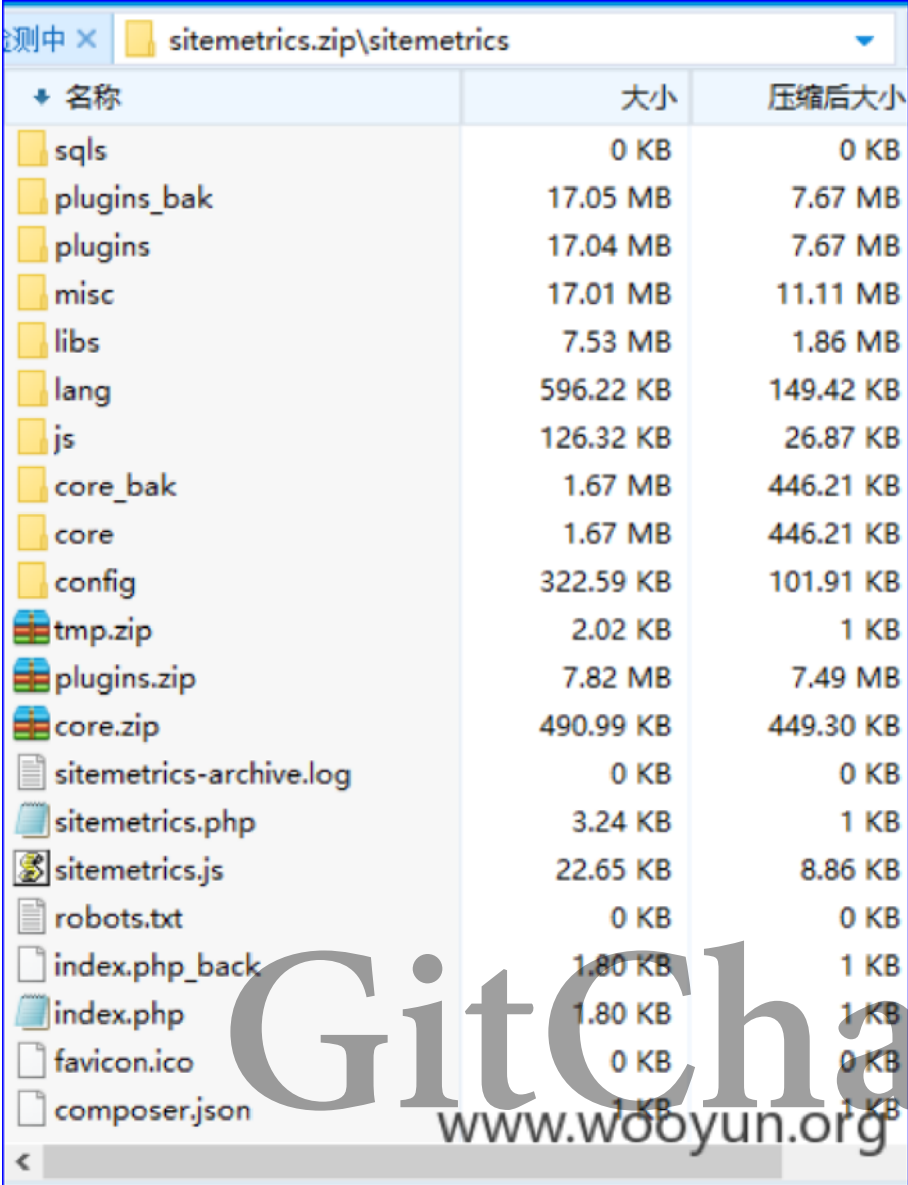
3.2 SVN源码泄露案例 案例

2015年10月，乌云白帽子提交漏洞“我爱我家某处源码泄露”。缺陷编号：wooyun-2015-0149331

在我爱我家有一处域名为 data.5i5j.com ,白帽子发现下面的地址可以访问到：

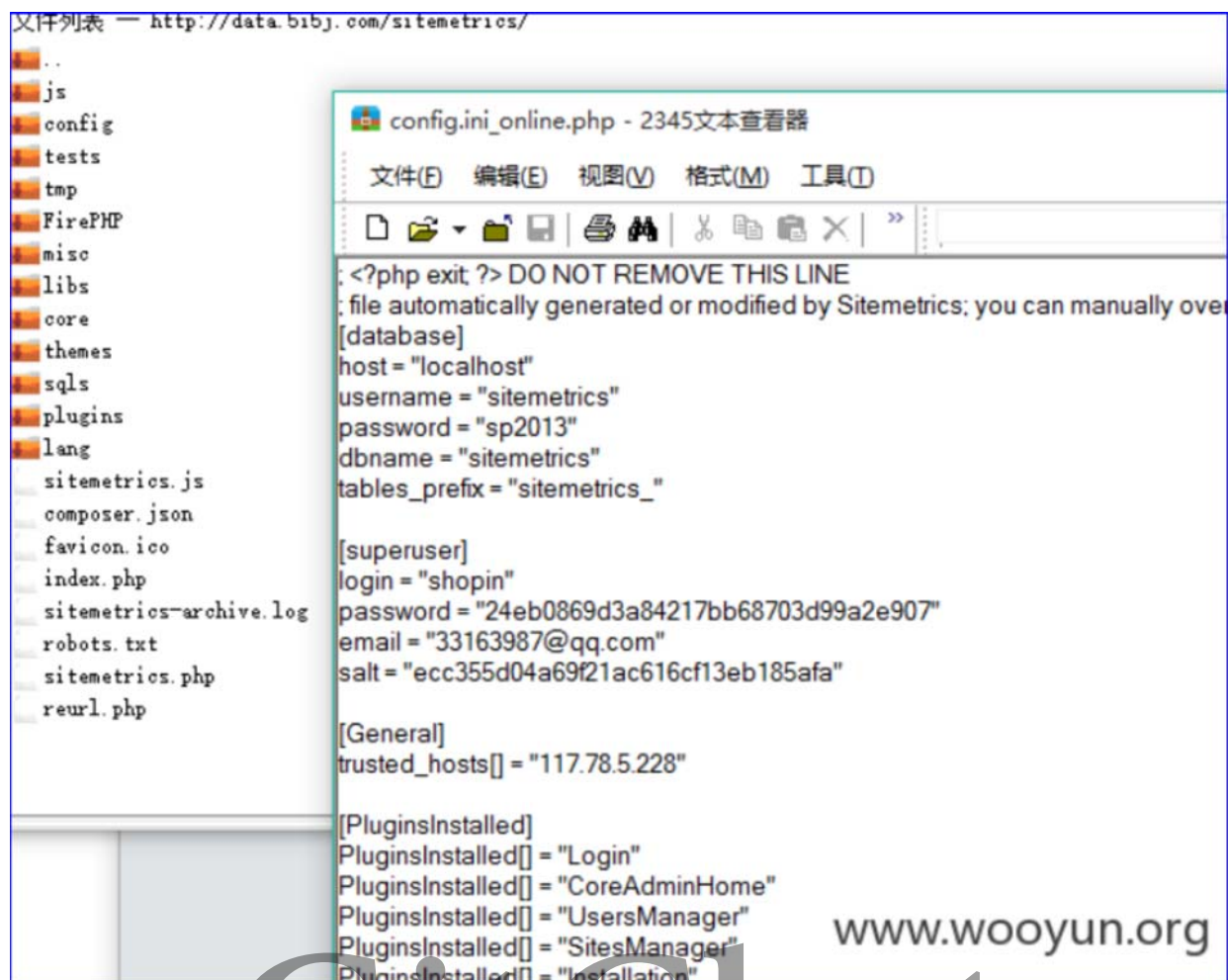
<http://data.5i5j.com/sitemetrics/.svn/entries>

白帽子知道使用svn版本控制器会在目录中生成.svn文件夹，于是猜测该处存在源码泄露问题，于是使用工具dvcs-ripper对其进行了一番验证，验证证实了最初的猜测，并得出了下面的目录以及代码文件。



名称	大小	压缩后大小
sqls	0 KB	0 KB
plugins_bak	17.05 MB	7.67 MB
plugins	17.04 MB	7.67 MB
misc	17.01 MB	11.11 MB
libs	7.53 MB	1.86 MB
lang	596.22 KB	149.42 KB
js	126.32 KB	26.87 KB
core_bak	1.67 MB	446.21 KB
core	1.67 MB	446.21 KB
config	322.59 KB	101.91 KB
tmp.zip	2.02 KB	1 KB
plugins.zip	7.82 MB	7.49 MB
core.zip	490.99 KB	449.30 KB
sitemetrics-archive.log	0 KB	0 KB
sitemetrics.php	3.24 KB	1 KB
sitemetrics.js	22.65 KB	8.86 KB
robots.txt	0 KB	0 KB
index.php_back	1.80 KB	1 KB
index.php	1.80 KB	1 KB
favicon.ico	0 KB	0 KB
composer.json	1 KB	1 KB

在文件目录中，发现一个文件名比较敏感，对其打开查看发现里面包含了数据库地址，用户名，密码等信息。



4. DS_Store文件泄漏

4.1 漏洞成因

.DS_Store文件 MAC系统是用来存储这个文件夹的显示属性的：比如文件图标的摆放位置。如果用户删除以后的副作用就是这些信息的失去。

这些文件本来是给Finder使用的，但它们被设想作为一种更通用的有关显示设置的元数据存储，诸如图标位置和视图设置。当你需要把代码上传的时候，安全正确的操作应该把 .DS_Store 文件删除才正确，因为里面包含了一些目录信息，如果没有删除，攻击者通过 .DS_Store 可以知道这个目录里面所有文件名称，从而让攻击者掌握了更多的信息。

在发布代码时未删除文件夹中隐藏的.DS_store，被发现后，获取了敏感的文件名等信息。攻击者可以利用访问URL（http://www.localhost.test/.ds_store）的方式来判断，是否存在 DS_store 泄露，如果存在泄漏，使用工具：dsstoreexp，就可以轻松的下载出源代码。

如下面的命令：

```
ds_store_exp.py http://www.localhost.test/.DS_Store
```

打开数据库文件对应的URL，在其中可以找到后台管理员账户和密码：


```
www.tcl-ectv.com/tcl_ectv.sql
访问最多  火狐官方网站  新手上路  常用网址

DROP TABLE IF EXISTS `tcl_admin`;
SET @saved_cs_client      = @@character_set_client;
SET character_set_client  = utf8;
CREATE TABLE `tcl_admin` (
  `id` mediumint(8) unsigned NOT NULL auto_increment,
  `typer` enum('system','manager','editor') NOT NULL default 'editor',
  `user` varchar(100) NOT NULL default '',
  `pass` varchar(50) NOT NULL default '',
  `email` varchar(100) NOT NULL default '',
  `modulelist` text NOT NULL COMMENT '可管理的模块, 系统管理员无效',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
SET character_set_client  = @saved_cs_client;

--
-- Dumping data for table `tcl_admin`
--

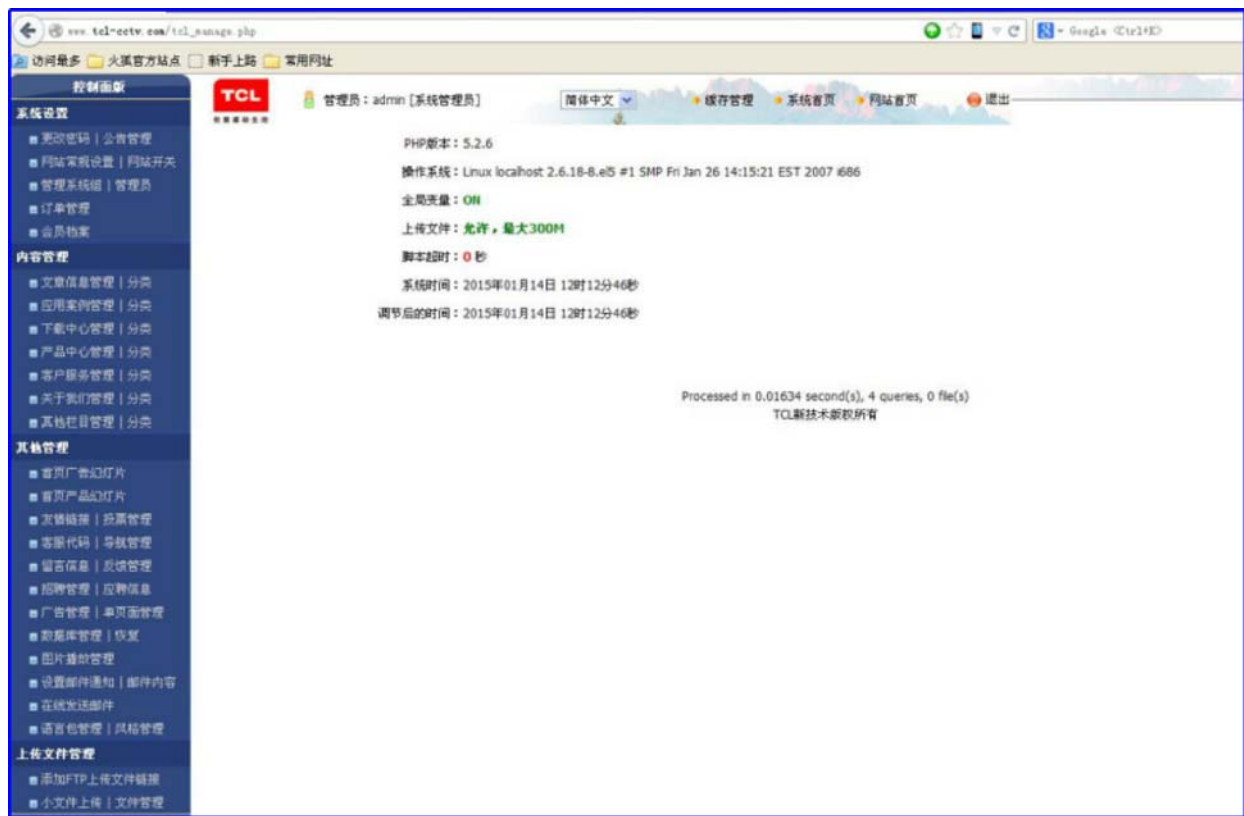
LOCK TABLES `tcl_admin` WRITE;
/*!40000 ALTER TABLE `tcl_admin` DISABLE KEYS */;
INSERT INTO `tcl_admin` VALUES (1,'system','admin','c5b5ae86dfccc8beefec','admin@admin.com','');
/*!40000 ALTER TABLE `tcl_admin` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `tcl_book`
--
```

用户名: admin 密码: c5b5ae86dfccc8beefec , 通过cmd5.com解密后, 可以得到真实的密码。



在后台URL中, 输入账号密码, 可以看见下图已经登录成功。



5. 网站备份压缩文件

5.1 漏洞成因

GitChat

在网站的升级和维护过程中，通常需要对网站中的文件进行修改。此时就需要对网站整站或者其中某一页面进行备份。

当备份文件或者修改过程中的缓存文件因为各种原因而被留在网站web目录下，而该目录又没有设置访问权限时，便有可能导致备份文件或者编辑器的缓存文件被下载，导致敏感信息泄露，给服务器的安全埋下隐患。

该漏洞的成因主要有是管理员将备份文件放在到web服务器可以访问的目录下。

该漏洞往往会导致服务器整站源代码或者部分页面的源代码被下载，利用。源代码中所包含的各类敏感信息，如服务器数据库连接信息，服务器配置信息等会因此而泄露，造成巨大的损失。被泄露的源代码还可能会被用于代码审计，进一步利用而对整个系统的安全埋下隐患。

`.rar` `.zip` `.7z` `.tar.gz` `.bak` `.swp` `.txt`

5.2 备份压缩文件 案例

2014年5月，乌云白帽子“N0xxx”提交漏洞“百度某分站备份文件泄露”，缺陷编号：wooyun-2014-050622

百度网盟的URL地址是：

`http://wm123.baidu.com`

白帽子无意中发现在URL加上域名+.tar.gz 也就是URL：

`http://wm123.baidu.com/wm123.tar.gz`

没想到就下载了网站源码；在其源码中还发现了数据库的链接地址，以及账号信息，如下图：



```
jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.cap.url=jdbc:mysql://.....:3006?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
jdbc.cap.read.db01.url=jdbc:mysql://.....:3006?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
jdbc.cap.read.db02.url=jdbc:mysql://.....:3006?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
jdbc.cap.username=beidoudb
jdbc.cap.password=cAnghAiYisHeNgxiAo

jdbc.xdb.url=jdbc:mysql://.....:106?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
jdbc.xdb.read01.url=jdbc:mysql://.....:3106?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull

jdbc.xdb.username=l.....
jdbc.xdb.password=c"....."

jdbc.maxPoolSize=10
jdbc.minPoolSize=5
jdbc.initialPoolSize=5
jdbc.idleConnectionTestPeriod=1800
jdbc.maxIdleTime=3600
```

www.wooyun.org

6. WEB-INF/web.xml泄露

6.1 漏洞成因

WEB-INF是Java的WEB应用的安全目录。该目录原则上来说是客户端无法访问，只有服务端才可以访问。如果想在页面中直接访问其中的文件，必须通过web.xml文件对要访问的文件进行相应映射才能访问。

WEB-INF主要包含一下文件或目录：

- /WEB-INF/web.xml：Web应用程序配置文件，描述了 servlet 和其他的应用组件配置及命名规则。
- /WEB-INF/classes/：含了站点所有用的 class 文件，包括 servlet class 和非servlet class，他们不能包含在.jar文件中
- /WEB-INF/lib/：存放web应用需要的各种JAR文件，放置仅在这个应用中要求使用的jar文件,如数据库驱动jar文件
- /WEB-INF/src/：源码目录，按照包名结构放置各个java文件。
- /WEB-INF/database.properties：数据库配置文件

不过在一些特定的场合却会让攻击者能读取到其中的内容，从而造成源码泄露。

6.2 WEB-INF目录配置漏洞 案例

2013年2月，乌云白帽子“Asuimu”提交漏洞“华为官网WEB-INF目录配置文件导致信息泄露”，缺陷编号：wooyun-2013-022906

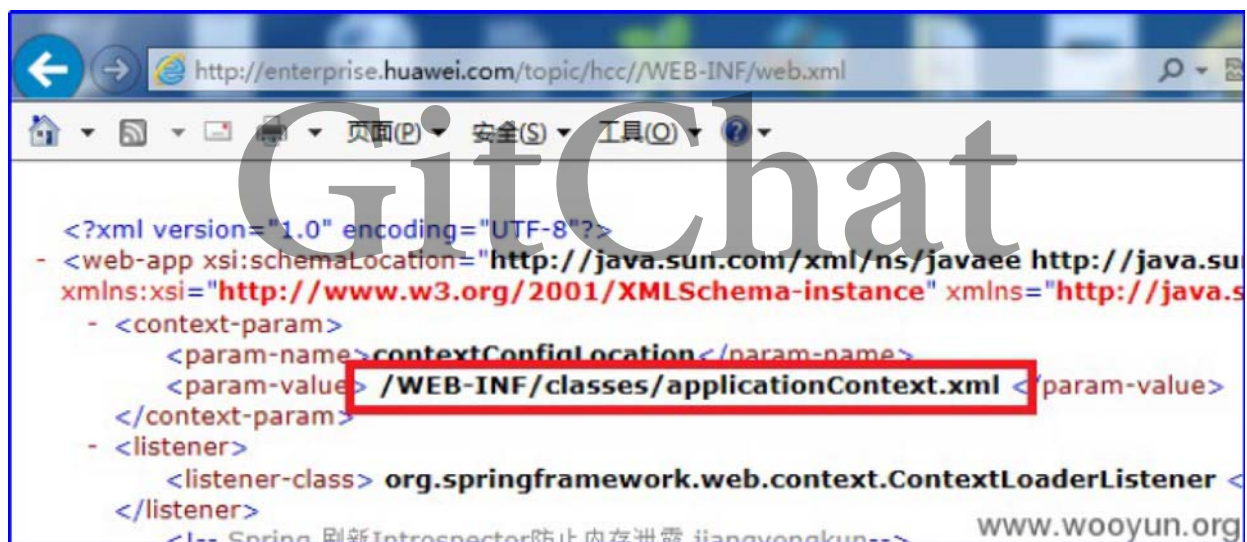
在该漏洞中由于目录权限未做好控制，导致网站配置信息泄露以及源码泄露问题。

此漏洞的WEB-INF目录位置URL为：

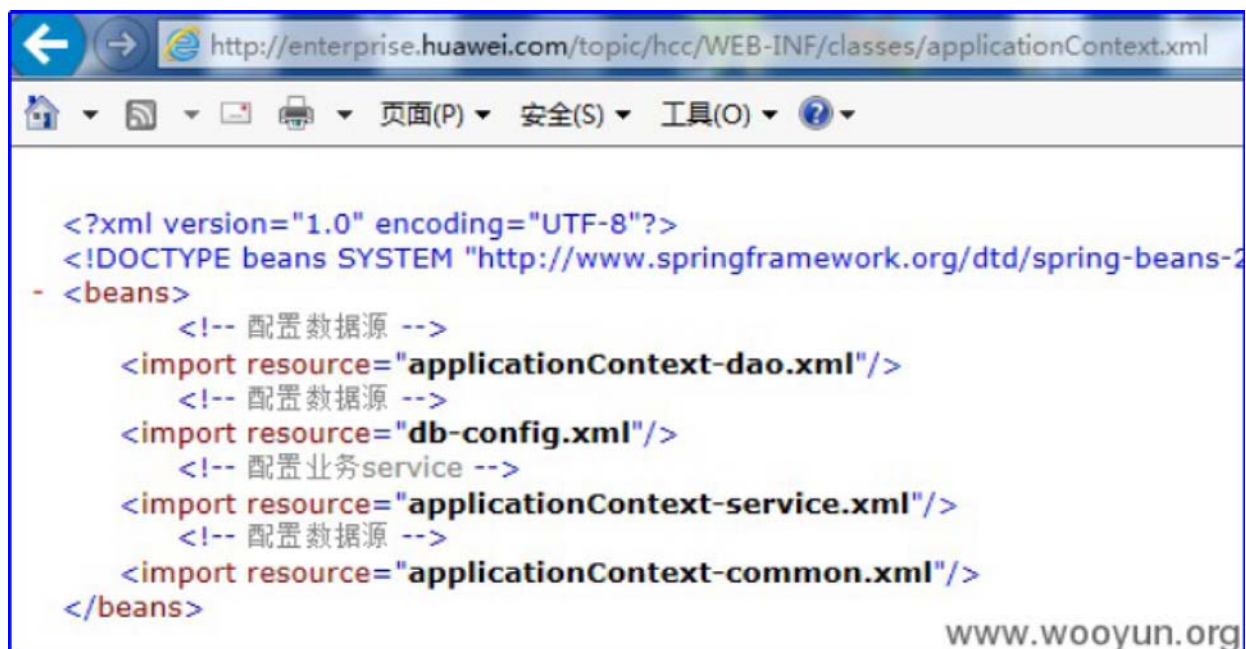
<http://enterprise.huawei.com/topic/hcc/WEB-INF/>

白帽子首先寻找配置文件（web.xml）的位置，通过web.xml的位置得到URL为：

<http://enterprise.huawei.com/topic/hcc/WEB-INF/web.xml>



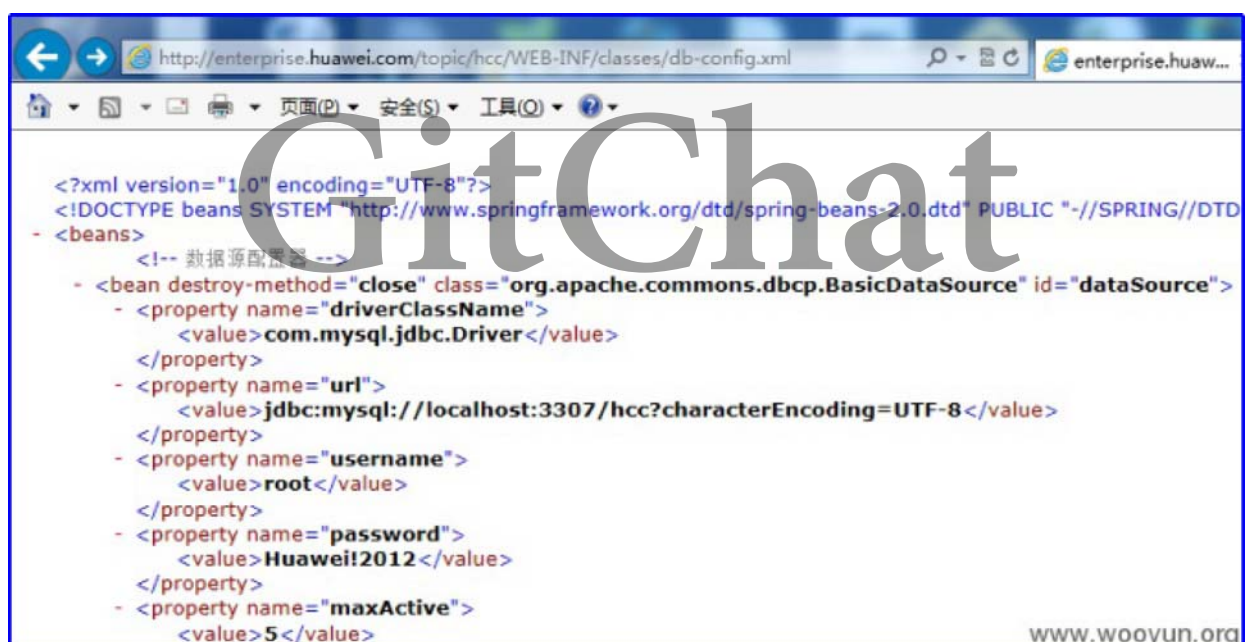
访问URL后，能看到如上图中的内容，白帽子发现有一个classes/applicationContext.xml文件；访问此文件对应的URL后，又从此文件中找到了数据库配置文件db-config.xml的路径。



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM "http://www.springframework.org/dtd/spring-beans-2
- <beans>
    <!-- 配置数据源 -->
    <import resource="applicationContext-dao.xml"/>
    <!-- 配置数据源 -->
    <import resource="db-config.xml"/>
    <!-- 配置业务service -->
    <import resource="applicationContext-service.xml"/>
    <!-- 配置数据源 -->
    <import resource="applicationContext-common.xml"/>
</beans>
```

www.wooyun.org

打开db-config.xml对应的URL后，能看到mysql的连接信息，比如 root Huawei!2012 localhost等等信息。不过因为数据库限制只能本地连接，所以白帽子并没有连接上数据库。



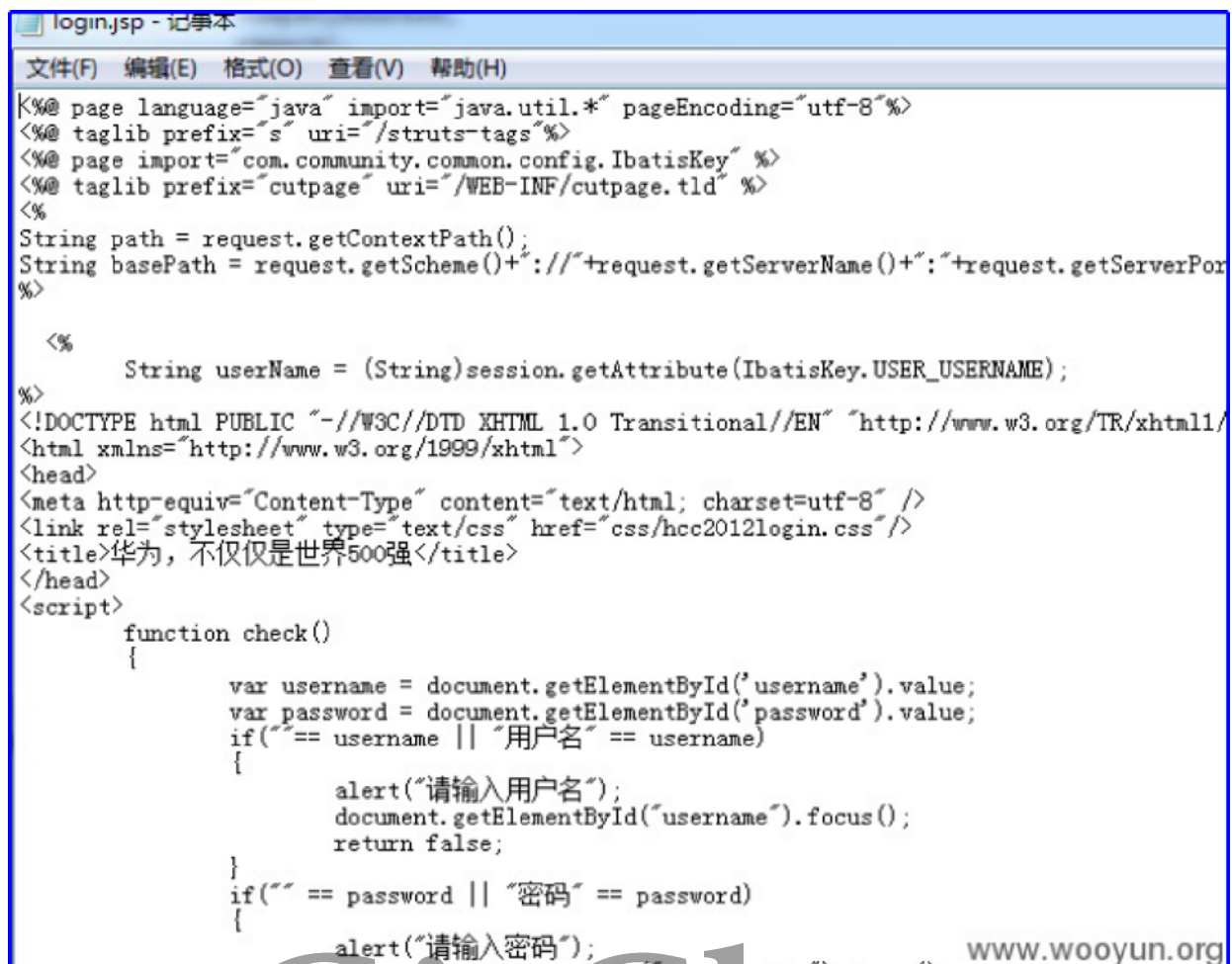
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM "http://www.springframework.org/dtd/spring-beans-2.0.dtd" PUBLIC "-//SPRING//DTD
- <beans>
    <!-- 数据源配置器 -->
    <bean destroy-method="close" class="org.apache.commons.dbcp.BasicDataSource" id="dataSource">
        <property name="driverClassName">
            <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
            <value>jdbc:mysql://localhost:3307/hcc?characterEncoding=UTF-8</value>
        </property>
        <property name="username">
            <value>root</value>
        </property>
        <property name="password">
            <value>Huawei!2012</value>
        </property>
        <property name="maxActive">
            <value>5</value>
        </property>
    </bean>
</beans>
```

www.wooyun.org

通过此漏洞还下载了部分源码，比如URL：

<http://enterprise.huawei.com/topic/hcc/login.jsp>

对应下图中的源码：



```
login.jsp - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ page import="com.community.common.config.IbatisKey" %>
<%@ taglib prefix="cutpage" uri="/WEB-INF/cutpage.tld" %>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+
%>

<%
    String userName = (String)session.getAttribute(IbatisKey.USER_USERNAME);
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/hcc2012login.css"/>
<title>华为，不仅仅是世界500强</title>
</head>
<script>
    function check()
    {
        var username = document.getElementById('username').value;
        var password = document.getElementById('password').value;
        if("" == username || "用户名" == username)
        {
            alert("请输入用户名");
            document.getElementById("username").focus();
            return false;
        }
        if("" == password || "密码" == password)
        {
            alert("请输入密码");
        }
    }
</script>

```

4. 防御方案

从上面的五种泄露方式可以看出，大部分情况都是代码上传后没有及时清理附带信息所造成的。

因此我建议代码发布尽量使用工具rsync来发布，因为此工具同步时可以排除一些目录或者文件，比如要排除所有.svn文件，可以如下面的命令行来排除，git同理。

```
rsync -avlh --exclude=*.svn root@192.168.1.100:~/tmp/
/data/version/test/
```

如果不能生产环境不能使用rsync，也给大家几点小建议：

1. Git 在仓库的根目录新建一个文件夹，把代码放入此文件夹中，网站的根目录应该指向此文件夹。这样攻击者就不能访问到.git文件夹的内容了。
2. 不要直接使用git或SVN等工具拉去代码到生产目录，可以在一个临时目录先拉去下来，把其中的一些版本控制器附带信息都去掉后再同步到生产目录。
3. 使用mac系统的开发者需要注意不要把.ds_store文件上传上去，因为里面包含在一些目录信息，会导致文件名称泄露

4. Web生产目录中不要存放代码压缩文件，这些文件极有可能被攻击者所发现，而下载下来。

GitChat