by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity  
**摘要**

谢各位捧场，已经打到第四关了。经过几次交流，发现SAE的童鞋对安全很重视，而且持“欢迎来搞”的态度，对这样的态度，给予肯定，有了这样的态度，作者才有了继续下去的欲望，我们都相信SAE最终会变得越来越安全。

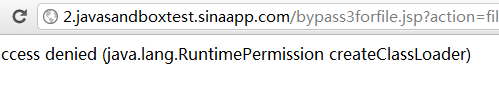
**正文**

前文《SAE云服务安全沙箱绕过3(绕过命令执行防御)》（http://www.inbreak.net/archives/426），作者利用了SAE仅仅限制EXEC命令，没有限制其他函数的漏洞，打破了整个沙盒的防御，最后给出的建议中，作者也提到了对沙盒的防御方式。SAE接受了常规的沙盒安全方案，不再局限于限制某个函数等策略。现在到处都是限制，这对于作者来说真是艰难困苦的开始，感觉SAE对于沙盒的安全限制，已经比google还要BT了，甚至有些不照顾JAVA框架了，无论如何，作者继续绕了下去。

**环境探测**  
首先看看常规的，危险的几个函数：

|  |
| --- |
| crackClassLoader |

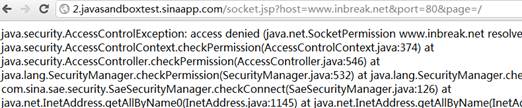
还是前几篇文章专用的代码，所以本文就不再次贴出来了，直接看结果：

[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass41.png)

已经限制了createClassLoader，这样我们自定义的classLoader已经无效，这意味着，不能再走定义一个自定义权限的类，这个流程了。

**连接socket测试**

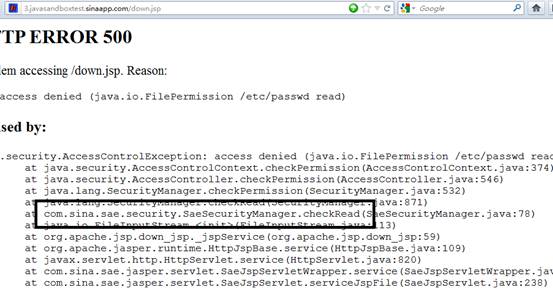
http://2.javasandboxtest.sinaapp.com/socket.jsp?host=www.inbreak.net&port=80&page=/

[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass42.jpg)

不允许socket连接，不允许XXX，不允许XXX。。。

这样绕，不是办法，都被干掉了。这时候，我们需要一种打破常规沙盒绕过的手段。

**读文件测试**

[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass43.jpg)

作者发现了一个非常有趣的情况，SAE的JAVA环境中，有个Class，叫做”com.sina.sae.security.SaeSecurity”，这个Class，像是做验证的，里面有方法“checkRead”。如果知道这个方法的验证逻辑，并且刚好开发人员实现这个方法时，出现了逻辑漏洞，才可能出现bypass。但是在这样的条件下，拿到SAE的源码，非常难，只能可以从侧面探测。

作者想到，如果让这个类，刚好在权限验证时，起效果，并且在java远程的沙盒之前run起来，最简单的架构，是让它直接继承Java.lang.SecurityManager，这个东西是java原生提供的功能，默认会根据java的policy文件中设置的权限白名单，做各种认证。一个Class继承了SecurityManager，就是所谓的自定义SecurityManager，SAE和google都使用了自定义SecurityManager。

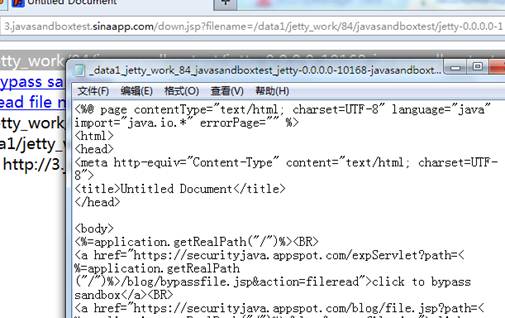
查看这个Class的文档说明，里面有很多权限验证的方法，这些验证方法，都可以由开发人员自行实现，覆盖java原生的验证逻辑。

[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass44.jpg)  
图中有很多验证方法，都可以由开发人员自己实现的验证方法覆盖，一旦SAE的开发人员覆盖了这些方法，比如命令执行，那么在真正java做命令执行前，会验证权限，就使用SAE开发人员写的方法，而不是JAVA原生的验证。

正常JAVA沙盒错误的信息，应该和SAE自己实现的安全认证没有关系，理论上java自己的沙盒，已经具备了文件读取等认证。但是按照前面图中所示的异常信息，作者推测，是SAE的开发人员替代了JAVA原生安全认证。从这里也看出，SAE野心不小，有替换JAVA语言本身安全沙盒的趋势，但是JAVA本身的代码，是经过很多年风吹雨打的，各种bypass都出现过，SAE新写的代码，想达到这个标准，还是比较难的。

下面可以从各种角度，测试SAE自己实现沙盒代码的安全性，checkRead方法，看名字就是验证文件读取权限的。而用户自己上传的文件，必然是可以读取的，否则沙盒的限制，就太BT了。

作者上传了一个文件，然后读取，获取文件路径的方式，见之前作者写的文章《SAE云服务安全沙箱绕过3(绕过命令执行防御)》（http://www.inbreak.net/archives/426）。

[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass46.jpg)

这是可以读的，这说明路径：

/data1/jetty\_work/84/javasandboxtest/jetty-0.0.0.0-10168-javasandboxtest.war-\_3\_javasandboxtest-any-/webapp

下的文件，都是可读的，那么文件权限验证，肯定就必须包含这个路径了。这个安全策略是否可以绕过呢？

**开始bypass**

这让作者想起了文件下载漏洞。绕过任意文件下载漏洞，有无数前辈的经验，最常见的思路，当然是“/../“了。

读取/etc/passwd文件内容：

|  |
| --- |
| http:*//3.javasandboxtest.sinaapp.com/down.jsp?filename=/data1/jetty\_work/84/javasandboxtest/jetty-0.0.0.0-10168-javasandboxtest.war-\_3\_javasandboxtest-any-/webapp/../../../../../../etc/passwd* |

一次成功！这种东西真的和运气，有很大的关系！  
[](http://www.inbreak.net/wp-content/uploads/2012/08/saebypass47.jpg)

从这个过程，也进一步验证了SAE的认证代码，checkRead方法，很可能只有一行，下面是推测出来的伪代码：

|  |
| --- |
| **If** (XXXX.startwith(/data1/jetty\_work/84/javasandboxtest/jetty-0.0.0.0-10168-javasandboxtest.war-\_3\_javasandboxtest-any-/webapp))  {  **Return** **true**;  } |

这种漏洞，本来就是个经典的文件下载漏洞代码，但是放到沙盒里，却导致了沙盒被绕过。现在我们又可以读取云上用户的文件了。

**关于沙盒与JSP空间友好性的问题**

我们知道JAVA的web应用，只有jsp和servlet的时代，已经不存在了，无论哪个应用，多少都会有JAVA框架的影子，比如struts、spring等等。框架本身的实现，经常会涉及到getClassLoader等等操作，现在SAE限制了这个东西，作者对STRUTS2应用，能否在SAE上跑起来，表示疑问。为此，作者专门上传了struts的官方showcase上去。  
结果返回503错误，不可用。

**总结**

即使是非常简单的STRUTS2应用都不可用，还有很多其他框架，也都有自己操作classLoader的情况，如果不支持JAVA框架，这样的SAE，还有啥存在的价值呢？所以这些权限，不止简单的禁止了事，它后面还有很多东西要做。没有可用性的安全，不是真的安全，这样的安全处理方案，早晚要在业务的压力下消失不见，要保证可用性，也要保证安全，这才是真正的安全。所以，坐等SAE对classLoader的限制消失，一切回归于0。

最好的办法，是对接管JAVA原生安全验证这部分代码，做一次整体的revuew，在代码无法保证安全之前，就去接管更加安全的东西，这本身就是一种安全隐患了，谁敢讲自己写的代码，一定比SUN（写沙盒的时候，还没到oracle）那帮人更安全，更健壮？

其实沙盒绕过技术，作者也是第一次玩，每次一定要等SAE修补之后，才会有新的办法出现，SAE肯定希望我一次性，把所有绕过的方法都给出来，但是问题是还没做好防御，谈何攻击呢？建议对SAE的代码，做一次整体的review，尤其是接管JAVA沙盒认证这块，相信还有更加严重的漏洞在里面，解决这个问题，只有由安全人员一行一行看了，如果SAE足够开放，建议直接开源出来，这可以更加开放的得到安全信息。  
by 空虚浪子心 http://www.inbreak.net 微博：http://t.qq.com/javasecurity