附件2

第一届中国研究生网络安全创新大赛揭榜挑战赛题目

**题目1：WAF防护有效性自动化评估**

**答疑邮箱： qingeng5@huawei.com**

**背景：**

给出具有漏洞的网站或靶机系统，含有web类高危漏洞，然后针对这个靶机系统前置流行WAF，要求挑战者针对WAF保护下的靶机展开自动化绕过尝试，最终输出可以绕过的Payload清单，并保障最终Payload可用作后续的渗透利用。

**竞赛题目详细描述：**

题面是一个专项web漏洞靶机系统（<https://github.com/zhuifengshaonianhanlu/pikachu>），该系统含有SQL注入、任意文件上传、任意文件下载/包含、RCE（命令注入、代码执行）主要高危漏洞。在该环境中依次部署5个WAF类产品，其中至少包含1-2个商用WAF（赛题评审前不提供具体WAF名称）。要求开发一套WAF规则自动化绕过工具，能够针对每个WAF保护下的各类漏洞做自动化绕过尝试，并最终输出可用的Payload。

**考察选手对WAF检测的认识以及编程能力，挑战内容如下：**

1、能够针对WAF保护下的高危漏洞（至少3个）进行自动化绕过尝试

2、 输出的绕过Payload，必须具备渗透利用价值，不能只是PoC

**评价方式：**

1、输出可利用的绕过型Payload列表

2、系统设计与WAF自动绕过算法

**题目2：网页登录报文自动化构造**

**答疑邮箱： qingeng5@huawei.com**

**背景：**

在对大量目标进行渗透测试过程中，使用各类web漏扫对发现的资产进行漏洞探测时，有相当多的资产是需要登录之后才能探测到漏洞的。传统的扫描器基本只能针对单一特定页面录制自动化登录脚本实施自动化渗透，需要开发一套漏扫爬虫数据自动化处理工具，能够实施批量化登录页面识别加自动化登录获得会话功能。

**竞赛题目详细描述：**

题面是500-1000个含有表单提交的交互页面地址，其中10%为登录页面地址。开发一套工具，能够导入这些地址后实施页面内容获取，并自动对登录页面进行识别和筛选，并在筛选完后使用预制的账户名和口令字典进行爆破。

**考察选手编程能力，挑战内容如下：**

1、表单提交页面地址必须含各类表单请求提交功能，相似功能不能超过10%

2、必须有一半为综合页面中含有登录功能，而非全部是登录入口页面
3、各个登录页面的表单请求参数必须完全不同

**评价方式：**

1、能够支持资产列表导入

2、输出登录表单准确率高于80%

**题目3：轻量级设备数据保护**

**答疑邮箱：l.dmxcsnsbh@gmail.com**

**背景：**

数据全生命周期安全保护很大程度上依赖设备本身的安全级别。针对没有安全环境的弱设备，如何保护数据全生命周期安全存在一定挑战。
**题目描述：**

小明购买了一个带屏的家庭存储盒子，并邀请家人一起在盒子里存储家庭相册、视频。家庭成员可以在设备上存储数据，并发起访问。



**问题：**

设计一个系统，保护轻量级设备上的数据。设计上包括以下维度：
1、数据at rest保护（必选设计）：
前提：在设备不具备基于安全环境的秘钥管理条件；
设计要素：如何保护数据存储的安全，使设备丢失的情况下，攻击者无法获取数据。
2、数据in use保护（必选设计）：
前提：设备不具备用户认证条件 （该设备没有锁屏密码、传统人脸、指纹等生物特征认证能力）
设计要素：对轻量级设备上存储的设备进行访问时，如何保证是合法的用户
3、其他要素（可选）：

为了保护该设备上的数据，还有哪些其他的考虑与防护目标？针对这些防护目标进行安全设计

**评估方式：**

输出设计报告，识别该场景下的安全模型与关键威胁，针对数据的机密性、完整性、访问方防仿冒、防DOS等维度进行分析设计。

**题目4：OpenHarmony漏洞挖掘**

**答疑邮箱：l.dmxcsnsbh@gmail.com**

**背景：**

OpenHarmony 是面向全场景、全连接、全智能的新型开源智能终端设备操作系统，旨在促进万物互联产业的繁荣发展。OpenHarmony 所涉及的设备相对复杂，上至高级智能终端，下至硬件资源极有限的传感器设备等，因此运用虚拟化仿真技术对其进行漏洞挖掘是主流且可行的方案。

**题目描述：**

利用 OpenHarmony 内核的仿真环境，并对其进行 Coverage-guided fuzz，发现尽可能多的漏洞。

**备注：**挑战者需了解常见内存破坏漏洞的类型和主要模式；了解主流操作系统内核漏洞的挖掘方法，并在仿真环境下针对一款主流操作系统（如Linux）的某一个模块（如文件系统）进行 fuzz。

**参考信息：**

https://gitee.com/openharmony

https://gitee.com/openharmony/device\_qemu

**评价指标：**

1、发现 OpenHarmony 的漏洞数量

2、Fuzzer 的覆盖率

**题目5：应用行为分析系统的设计与实现**

**答疑邮箱：l.dmxcsnsbh@gmail.com**

**背景：**

应用行为分析系统，是针对风险应用和恶意应用的重要手段，目前业界Android平台的应用行为分析工具大多基于Xposed插件、Frida动态插桩、Magisk插件或者应用多开等技术实现。然而针对这些已有的实现手段，在产业链中已经有较为成熟的方法进行检测，并且由于这类插件大多需要取得root权限并修改大量的系统文件和策略，导致很难在根本上杜绝被第三方应用检测到。基于这样的背景，本课题提出一种更隐蔽的方法来实现应用行为分析系统。

**题目描述：**

设计一套应用行为分析系统，可以检测风险应用和恶意应用的行为，至少包含以下需求：

1、基于修改的AOSP源代码，实现第三方应用调用Binder系统服务的记录，并且能将数据存储到本地数据库和导出到PC上进行进一步分析。

2、基于修改的AOSP源代码，实现第三方应用调用Linux Kernel Syscall的记录，并且能将数据存储到本地数据库和导出到PC上进行进一步分析。

以下需求可供参考：

1、将收集到的数据传输到PC上之后，可以生成基于Web的数据看板，并列出相关的风险。

2、实现你认为应当记录的，对于风险应用和恶意应用行为分析有帮助的其他数据。

**评价指标：**

1、应用行为分析系统对第三方应用的透明度，也就是第三方应用进行检测对抗的难度，检测的技术包括但不限于root检测、ADB调试检测、模拟器检测、SELinux策略检测、关键进程检测等。

2、应用行为分析系统记录内容的全面性，至少需要包含调用Binder系统服务和调用Linux Kernel Syscall的记录，能记录更多数据为加分项。

3、对Android大版本的兼容性以及分析系统的可移植性，要求必须兼容AOSP 12。

**题目6：恶意脚本相似性问题**

**答疑邮箱：george.liu@163.com**

**背景：**

当前恶意脚本在主机安全领域造成非常大的危害，在脚本中做恶意软件下载、隐藏、横向扩展等。执行后，对系统造成极大破坏，除了植入恶意程序，造成的系统破坏也经常很难清理和恢复。检测困难，因为攻击者会对脚本做少量或大量变化，如其中的IP、用户名、目录名等，hash签名就发生了变化，使用签名hash匹配的方式无法检测。因为对文件进行文字比较会导致运算量很大，比较方法也很复杂，实际上不可行。所以，需要用相似性技术来检测文件是否是恶意脚本。

在这个题目中，是开发一个相似性算法，该算法处理恶意脚本可能经历的各种更改。由于真正的恶意脚本可能会造成危害，在这个题目中，使用文本文件模拟处理的恶意脚本。本题目中，我们使用“Locality-sensitive hashing” 位置敏感哈希，这个术语解释中的第一段作为基准文本。

为了模拟脚本可能经历的类似变化，对文本做了以下2种类型的变化：

—在文件中加入了一个文本块

—对文件中某些单词的变化——例如，变化一些单词，省略一些单词

**（1）基准文本：**

“In computer science, locality-sensitive hashing (LSH) is an algorithmic technique that hashes similar input items into the same "buckets" with high probability.[1] (The number of buckets is much smaller than the universe of possible input items.)[1] Since similar items end up in the same buckets, this technique can be used for data clustering and nearest neighbor search. It differs from conventional hashing techniques in that hash collisions are maximized, not minimized. Alternatively, the technique can be seen as a way to reduce the dimensionality of high-dimensional data; high-dimensional input items can be reduced to low-dimensional versions while preserving relative distances between items.”

**（2）增加一段文字之后的文本：**

“In computer science, locality-sensitive hashing (LSH) is an algorithmic technique that hashes similar input items into the same "buckets" with high probability.[1] (The number of buckets is much smaller than the universe of possible input items.)[1] Since similar items end up in the same buckets, this technique can be used for data clustering and nearest neighbor search. It differs from conventional hashing techniques in that hash collisions are maximized, not minimized. Alternatively, the technique can be seen as a way to reduce the dimensionality of high-dimensional data; high-dimensional input items can be reduced to low-dimensional versions while preserving relative distances between items.

The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for gauging the similarity and diversity of sample sets. It was developed by Grove Karl Gilbert in 1884 as his ratio of verification (v)[1] and now is frequently referred to as the Critical Success Index in meteorology.[2] It was later developed independently by Paul Jaccard, originally giving the French name coefficient de communauté,[3] and independently formulated again by T. Tanimoto.[4] Thus, the Tanimoto index or Tanimoto coefficient are also used in some fields. However, they are identical in generally taking the ratio of Intersection over Union. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets”

**（3）修改过一些文字的文本**

“In computer ***program***, locality-sensitive hashing (LSH) is an algorithmic technique that hashes similar input items into the ***similar*** "buckets" with high probability.[1] (The number of buckets is much ***larger*** than the universe of possible input items.)[1] Since similar items end up in the same buckets, this technique can be used for data clustering and nearest neighbor search. It differs from conventional hashing *methods* in that hash collisions are maximized, not minimized. Alternatively, the technique can be seen as a way to reduce the ***dimension*** of high-dimensional data; high-dimensional input items can be ***induced*** to low-dimensional versions while preserving relative distances between items.”

**目标：**

参赛者需要开发一种方法来检测此类文件之间的相似性水平，并说明解决方案。算法中考虑到算法复杂度、可扩展性等方面。**参赛者也可以使用自己准备的恶意脚本用例完成题目。**

这个问题是开放式的，但是有一些可能的方向可以作为参考：

—局部敏感哈希方法，例如SSDeep、TLSH等

—机器学习方法

—数据预处理方法，如文件分块等技术

**题目7：数字水印与隐藏信息**

**答疑邮箱：george.liu@163.com**

**背景：**

数字隐藏水印将水印藏入图片，也有将数据（秘密）隐藏在另一个公开数据中，让第三方无法发现和破坏。例如，**在图片中隐藏一个python代码文件**。

**题目：**

在这个题目中，用户A和用户B通过未加密的网络通道在两个网络之间传送数据，该通道被全天候监控。A希望向B传输一个文本文件，要让监控系统不会注意到它。他需要将文本文件嵌入灰度（2D）图像中。这个挑战的目标是在最小的迭代次数中传输最大数量的信息（即，最小化通过网络信道发送的图像的数量）

在图像中嵌入数据时，需要考虑三个方面：

1、安全性——指“不可注意”或“不可检测”——人类的“肉眼”不应该能够检测到数据被隐藏的事实。即，肉眼观察原始图像和隐藏信息的图像的比较不应不同

2、容量——使用多少个图像。在我们的用例中，图像和格式是确定的。

3、健壮性——如果修改（转换、压缩、旋转、缩放、调整大小等），隐藏信息需要能够被恢复，如把文件格式转换成jpeg格式。

在这个比赛中，是用最小的图片数量传输这个文本文件

1、图片是800\*600jpeg格式的黑白图片

2、要嵌入的秘密是一个python脚本文件，secret.py, 4.77KB。

**题目原则：**

1、**图片大小像素数不可改变，文件大小增加不超过5%**

2、数据可以完全恢复

3、不限制嵌入方法

**评估：**

1、使用的图片数量

2、查看原始图像和隐藏信息图像之间的差异

3、健壮性：把图像格式改为Png，对图像进行缩放之后，图像是否不可用

secret.py点击链接下载

<http://cpipc.acge.org.cn/sysFile/downFile.do?fileId=eac6a9d6bafe4e8e8523bba8632e2096>

用于隐藏水印的灰度图片点击链接下载

<http://cpipc.acge.org.cn/sysFile/downFile.do?fileId=92d2dd21f70e42e2bd3ea4d001357d0e>