北京大学

信息安全工程课程项目

项目报告

成　　员： 许志鹏 李子康 陈菊芳

院　　系： 软件与微电子学院

导师姓名： 孙惠平

二〇一七 年 十 二 月

目录

一．项目背景

CAPTCHA 是“Completely Automated Public Test to tell Computers and Humans”(自动区分计算机和人类的图灵测试)缩写。验证码最初提出的目的是为了解决垃圾邮件问题。路易斯等人在卡内基梅隆第一次提出了 CAPTCHA(验证码)这样一个程序概念。这个程序基于这样一个重要假设：提出的问题要容易被人类解答，并且让机器无法解答。在当时的条件下，计算机辨识技术还很落后，识别扭曲的图形，对于机器来说还是一个很艰难的任务，而人类却可以轻松的认出这些文字。这是一个简单而巧妙的设计，计算机先是产生一个随机的字符串，然后用程序把这个字符串的图像进行随机的污染，扭曲，再显示给显示器前的人或者机器。凡是能够辨识这些字符的，即为人类。

验证码能够防止网站恶意注册、防止恶意刷票、防止暴力密码破解、防止论坛垃圾广告信息。有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试，目前验证码是现在很多网站通行的方式（比如招商银行的网上个人银行，腾讯的 QQ 社区），是当前每一个网站不可回避的设计，随着验证码的普及与发展，最初设计的文本验证码演变处一些新的形式，如目前国内部分商家使用的汉字验证码。验证码主要是运用于登录，注册，评论发帖等网站模块。

1) 防止网站恶意注册。许多机器人程序能自动注册账号，然后发布消息。使用验证码可以使得只有人能够发布信息而阻止机器人程序发布信息，这样有效阻止了病毒或垃圾信息横行，让用户能够更有效的利用互联网资源。
2) 防止恶意刷票。一个自动注册程序可以自动批量注册上千万的账号，继而到指定网站上刷票，使用验证码可以有效的防止恶意刷屏。
3) 防止暴力破解。盗号程序可以利用字典攻击尝试得到密码，若使用验证码，则能大大的降低暴力破解密码的可能性，提高互联网的安全性。
4) 防止论坛垃圾广告信息。机器人能自动登陆论坛发布垃圾信息，使用验证码则可以有效的阻止机器人自动发布垃圾信息。避免造成网络垃圾信息泛滥。

全球通行的字母、数字验证码是目前最为广泛使用的一种较为简单的算法，也是相对访客而言较为人性化的简单设计之一。Web 安全的攻击与防护技术一直是互相促进的，验证码识别技术的不断发展推动着验证码生成技术的提高。

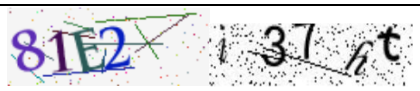
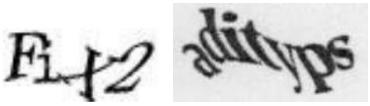
二．产品现状

许多网站使用验证码来试图阻止与它们站点的自动交互，例如网站注册、网上密码找回、投票系统、在线交易、订票系统、信息发布、社交网络登录验证等环节，系统要求登录者必须输入验证码图片上所显示的数字或字母或中文字符才可以完成登录，以防止有人利用计算机机器人程序的自动大量登录社交网站账号或注册网站账号发布病毒或是垃圾广告。这些努力对于这些网站的成功是至关重要的，可以说，互联网处处都存在验证码，我们的互联网生活也离不开验证码。例如，Gmail 通过阻止自动垃圾邮件制造者的接入来提高它的服务；eBay 通过阻断来自诈骗网站的机器人来改进它的市场；Facebook 制用于滥发邮件给诚实用户及在游戏中欺骗的配置文件的产生。最广泛使用的验证码将扭曲的字符与模糊技术相结合，这样人类识别容易但自动识别困难。

基于字符文本的验证码是现今互联网上使用最多的验证码，也是本文研究破解的验证码。这种验证码一般提供一张图片，图片中有数字或者大小写字母，为了加大字符提取分割和机器识别难度，往往在图片背景中加入点噪声、线噪声、复杂背景噪声，图片中的字符也可能空心、旋转、扭曲、粘连、坍塌、重叠和干涉线等。（使用此类方式的主要商家有：百度和 eBay、qq 邮箱、中国工商银行，中国农业银行，中国银行，百度网盘等等），也有的字符文本验证码中含有中文字符，目前主要出现在国内，由于中国人的母语是汉语，且汉字验证码对于机器的自动识别的抵抗能力要比普通字母和数字要好的多（常用汉字有 3000 个，字母只有 26 个），这使得汉字顺利成章的被应用到验证码中，但是中文验证码往往用户体验较差，因为需要的键盘输入更多，为改善这一状况，许多使用中文验证码的网站在选取中文字符时不再随机选取中文文字，而是选取一个成语，因为这样可以带来较少的键盘输入。

在当今各大网站上，使用最多的验证码机制就是基于文本的验证码。它的实现比较容易，人类识别起来很方便，对人群素质要求低。文本验证码图片中的字符主要是中英文字符或者是数字。为了提高机器程序的识别难度，验证码生成系统会在背景中加入其它干扰和噪声。

下表是网络上面常见的文本验证码的设计特点及其应用系统。

验证码特点	举例说明	应用网站
干扰线、背景噪声		湖南大学教学服务系统网站 北京大学学生网上服务中心网站
复杂纹理背景	 	这个我还没有找到有哪些网站使用了
变换字体		北京大学选课系统 （区分大小写）
字符扭曲、旋转		支付宝网页登陆 Microsoft 注册验证码
字符粘连	 	北京大学选课系统 Google 注册验证码 Yahoo
空心验证码	 	雅虎、腾讯、新浪、中国移动、百度
中文验证码	 	百度的门户网站登录、人人网

光学字符识别技术（OCR）对无粘连、无旋转、无扭曲和少量离散噪声干扰的简单验证码识别上已经取得了比较好的效果。EZ-Gimpy 和 Gimpy 两种相似类型的验证码

已被 Mori 和 Malik 利用复杂对象识别算法进行了成功的识别，利用这种算法对 EZ-Gimpy 验证码的识别率可以高达 92%。Chellapillar 和 Patrice 从网站上获取的验证码，然后利用机器学习方法进行破解，识别率从 4.89%至 66.2%不等。2011 年，Elie Bursztein,Matthieu Martin 和 John C Mitchell 针对于常用网站采用 15 种不同类型的验证码机制利用系统评价分析发现其中 13 类验证码机制 很容易被电脑程式自动识别的。

在国内，验证码机制也在大量的网站中得到广泛应用，但是大部分的网站的验证码机制设计的都比较简单，对验证码的重视的程度不够。由于 CAPTCHA 机制的诸多问题还没有得到较好的解决，因此实际应用中绝大多数 CAPTCHA 都是不安全的。目前还缺少安全性和可用性都非常好，尤其理论上证明无法被破解的 CAPTCHA 机制。新的安全的 CAPTCHA 机制不断涌现，针对这些新的验证码机制的破解方法也断不断提出，网络安全的重要性越来越亟待提高，不管谁是最后的赢家，验证码机制将得到一个井喷式的发展，验证码机制的研究都是一个双赢的局面。

三．项目设计

1. 设计对象

验证码 CAPTCHA 是“Completely Automated Public Test to tell Computers and Humans”(自动区分计算机和人类的图灵测试)缩写，它的最初提出是为了解决垃圾邮件问题，现在主要用来防止恶意注册网站、恶意刷票、暴力破解密码和论坛垃圾信息。目前市面上的验证码有多种多样的形式，包括文本验证码、图片验证码、滑动验证码、声音验证码、指纹识别和人脸识别等。其中，基于字母和数字的文本验证码是如今互联网上使用最为广泛的验证码，也是本文的设计对象。

2. 设计目的

文本验证码一般提供一张图片，图片中有数字或者大小写字母，如图 2.1 所示。为了增加机器识别的难度，同时考虑用户识别的准确性，往往在图片中加入复杂的纹理背景和干扰线，图片中的字符也可能呈现空心、旋转、扭曲和粘连等样式。本项目的设计目的就是生成这样的一张验证码图片，并且我们自己能够对验证码的具体样式进行调控，使其展现不同的视觉效果和识别难度，另外，我们还将对不同样式的验证码图片进行简单的识别攻击，来具体分析验证码的不同样式对其识别难度的影响。

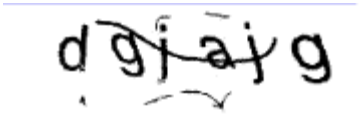


图 2-1 文本验证码

3. 设计思路

验证码的验证机制如图 3-1 所示，首先用户提交表单,服务器端收到用户请求开始会话（Session）,生成验证码图片时候将字符适当旋转并添加一定量的干扰线，每次随机选择一个数字或者字母共选取 4 个作为验证码长度，同时生成验证码字符串储存在内存中，将图片发送给用户，用户通过 Http 请求获得验证码图片，识别并填写提交表单发送到服务器，服务器将用户填写的验证码与内存中真实的验证码比对，相同则验证码成功，否则验证失败，刷新验证码。

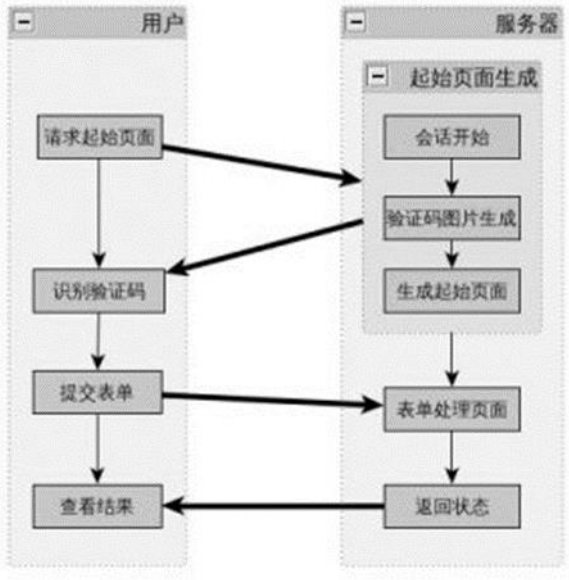


图 3-1

3.1 普通验证码的生成

a）使用 BufferedImage 用于在内存中存储生成的验证码图片

b）使用 Graphics 来进行验证码图片的绘制，并将绘制在图片上的验证码存放到 session 中用于后续验证

c）最后通过 ImageIO 将生成的图片进行输出

d）通过页面提交的验证码和存放在 session 中的验证码对比来进行校验

3.2 Kaptcha 验证码的生成

在本项目中，我们使用的是来源于 Google 的开源的验证码工具 Kaptcha，通过对其进行配置，我们可以生成各种样式的验证码。以下是我们的设计思路：

a) 新建 web 项目，导入 kaptcha-2.3.2.jar

b) 配置 web.xml 添加 servlet

c) Jsp 页面添加表单和验证码

d) 新建 Java 文件进行表单和验证码验证

e) 设置点击图片更换验证码

f) 修改 web.xml 更改图片样式

3.3 验证码的攻击

当前验证码识别技术的三个主要步骤是：第一，先进性预处理，将字符和背景分开。第二，对预处理后的图像进行分割，得到单个字符。第三，就是对单个字符进行识

别。预处理是指在对图像进行分割、特征提取和识别前所进行的处理。预处理的主要目的是对一个给定的含有字符文本的图像，突出图像中与字符有关的信息，削弱或去除某些不需要的信息。预处理可以采用图像处理领域的一些现有算法包括灰度化、二值化、去燥、归一化和细化等进行处理。验证码文本如果不是粘连的，即使加上很复杂的背景，也很容易被识别。所以 google 和 Yahoo 等几个公认比较安全的验证码机制都是使用了粘连的验证码来增加安全性。对于这类粘连的验证码识别，很重要的一步就是将粘连的字符切分开。传统的图像分割技术基本上都是将一个与背景的纹理、形态等特征有较大区别的目标，使用聚类、图论、水平集、多尺度理论、纹理特征提取等技术来实现目标图像与背景轮廓分割。针对单个字符的识别，当前的主要方法包括基于形状上下文、BP 神经网络、卷积神经网络、支持向量机等机器学习方法。

我们采用 pytesser 对简单验证码进行识别，pytesser，OCR in Python using the Tesseract engine from Google 是谷歌 OCR 开源项目的一个模块，可将图片中的文字转换成文本（英文和数字），光学字符识别(OCR,Optical Character Recognition)是指对文本资料进行扫描，然后对图像文件进行分析处理，获取文字及版面信息的过程。Tesseract 的 OCR 引擎最先由 HP 实验室于 1985 年开始研发，至 1995 年时已经成为 OCR 业内最准确的三款识别引擎之一。然而，HP 不久便决定放弃 OCR 业务，Tesseract 也从此尘封。数年以后，HP 意识到，与其将 Tesseract 束之高阁，不如贡献给开源软件业，让其重焕新生。2005 年，Tesseract 由美国内华达州信息技术研究所获得，并求诸于 Google 对 Tesseract 进行改进、消除 Bug、优化工作。Tesseract 目前已作为开源项目发布在 Google Project。PyTesser 使用 Tesseract OCR 引擎，将图像转换到可接受的格式，然后执行 tesseract 提取出文本信息。

四．项目实现

1． 普通验证码的生成

（a）使用 BufferedImage 用于在内存中存储生成的验证码图片

```
BufferedImage img = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
```

（b）使用 Graphics 来进行验证码图片的绘制，通过 java.awt.Graphics 类绘制，设置长宽、背景颜色、字体

```
int width = 120;
int height = 37;
int lines = 7;
BufferedImage img = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
Graphics g = img.getGraphics();
// 设置背景色
g.setColor(Color.WHITE);
g.fillRect(0, 0, width, height);
// 设置字体
g.setFont(new Font("黑体", Font.BOLD, 22));
```

(c)设置干扰线

```
// 干扰线
for (int i = 0; i < 8; i++) {
    int x1 = r.nextInt(width);
    int y1 = r.nextInt(height);
    int x2 = r.nextInt(width);
    int y2 = r.nextInt(height);
    g.setColor(new Color(r.nextInt(200), r.nextInt(200), r.nextInt(200)));
    g.drawLine(x1, y1, x2, y2);
}
```

(d）随机旋转

```
//旋转
Graphics2D g2d = (Graphics2D) g;
g2d.rotate(Math.PI * (r.nextInt(180) / -180));
AffineTransform trans = new AffineTransform();
trans.rotate(r.nextInt(10) * Math.PI / 180);
g2d.setTransform(trans);
g2d.drawString("" + s, 5 + i * width / 4, y);
```

2． Kaptcha 验证码的生成

a） 新建 web 项目，导入 kaptcha-2.3.2.jar

b） 配置 web.xml 添加 servlet

```
<servlet>
  <servlet-name>Kcaptcha</servlet-name>
  <servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class>
```

```
    </servlet>
    <servlet-mapping>
        <servlet-name>Kcaptcha</servlet-name>
        <url-pattern>/random.jpg</url-pattern>
    </servlet-mapping>
```

c) Jsp 页面添加表单和验证码

```
<form action="submit.action" method="post">
用户名: <input type="text" name="username"><br/><br/>
密   码: <input type="password" name="password"><br/><br/>
验证码: <input type="text" name="checkcode">  <img src="random.jpg" alt="验证码图片" align="center" title="看不清？点击更换图片" onclick="javascript:this.src='random.jpg?tm='+Math.ran
<input type="submit" value="提交"/>   <input type="reset">
</form>
```

d) 新建 Java 文件进行表单和验证码验证

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    String username = request.getParameter("username");
    String password = request.getParameter("password");
    String checkcode = request.getParameter("checkcode");
    String key = (String)request.getSession().getAttribute(com.google.code.kaptcha.Constants.KAPTCHA_SESSION_KEY);
```

e) 设置点击图片更换验证码

```
' onclick="javascript:this.src='random.jpg?tm='+Math.random()" />
```

f) 修改 web.xml 更改图片样式

3. 利用 Python 模块 pytesser 识别验证码

3.1 pytesser 源码

查看 pytesser.py 的源码可以看到下面两个方法：

(1)call_tesseract(input_filename, output_filename)

该函数调用 tesseract 外部执行程序，提取图片中的文本信息

(2)image_to_string(im, cleanup = cleanup_scratch_flag)

该函数处理的是 image 对象，所以需用使用 im = open(filename)打开文件，返回一个 image 对象。其中调用 util.image_to_scratch(im, scratch_image_name)将内存中的图像文件保存为 bmp，以便 tesserac 程序能正常处理。

3.2 pytesser 使用

在代码中加载 pytesser 模块，获取 img 目录下的全部验证码图片，调用 pytesser 的 image_to_string 方法读取验证码图片中的文本输出。

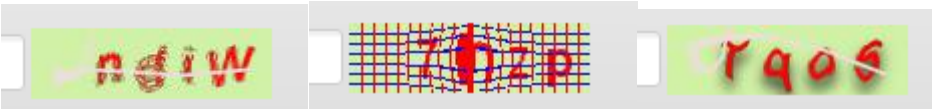
```
allFile = os.listdir('img')
for i in allFile:

    im = Image.open('img/'+i)
    text = pytesser.image_to_string(im).strip().lstrip()
    print text
```

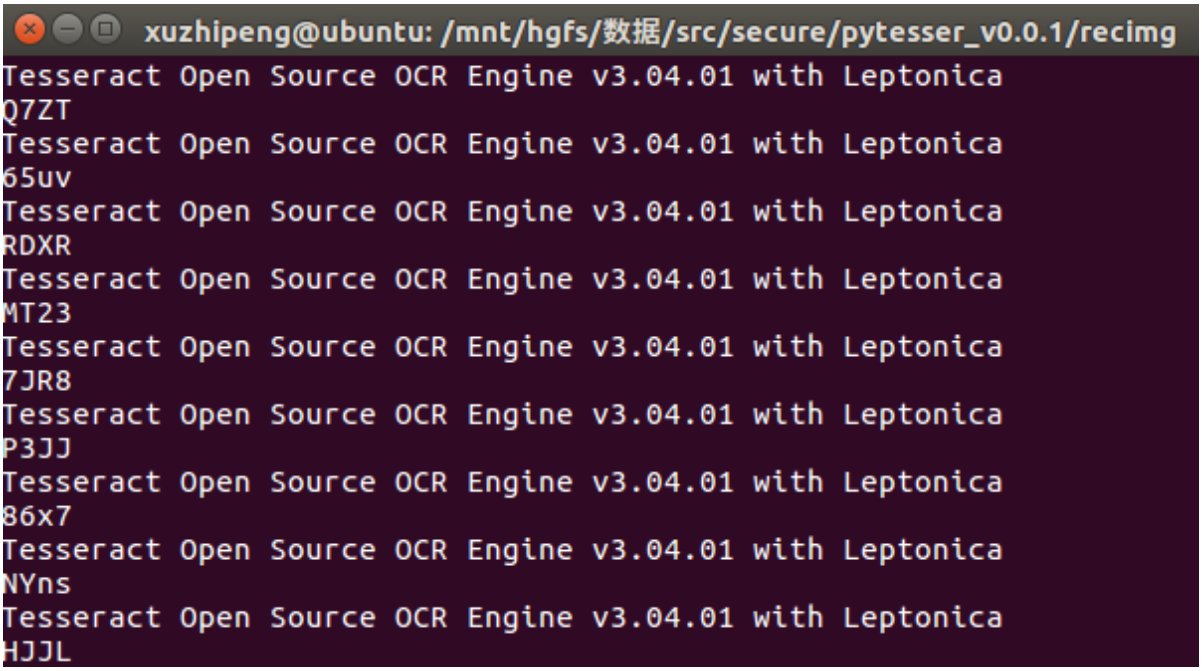
五．项目运行结果

1． 普通验证码的生成

2． Kaptcha 验证码的生成



3. Pytesser 识别验证码


xuzhipeng@ubuntu: /mnt/hgfs/数据/src/secure/pytesser_v0.0.1/recimg
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Q7ZT
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
65uv
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
RDXR
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
MT23
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
7JR8
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
P3JJ
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
86x7
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
NYns
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
HJJL

六．项目总结

基于数字和字母的文本验证码生成迅速、易于操作，的确是一种很简单通用的验证码。但也正因如此，它的识别难度要明显低于诸如滑动验证码、语音验证码等形式的验证码，在一些安全性要求很高的网站上不值得使用。假如我们已经选择了文本验证码，就有必要丰富它的样式来提升识别难度，具体则可以考虑背景颜色、字符颜色、字符样式（扭曲、空心、粘连等）和干扰线等因素。

项目组首先采用普通的验证码生成方式即用 java 的图形类 drawString 方法直接在图形上随机生成数字或字母，对生成的 1.0 版本进行 pytesser 模块识别，发现识别率达到惊人的 100%，意识到验证码需要采用一些策略防止识别，随后对验证码随机添加干扰线，pytesser 的识别率迅速下降，采用一定的旋转后，识别率进一步降低。此时又觉得识别率过于低，后查阅论文文献发现大多数验证码的攻击并不是简单的采用 tesseractOCR，而且还要对验证码不同的反识别反分割策略做相对应的改进。最后使用了开源的 kaptcha 开源验证码组件，发现效果很好，肉眼甚至都难以分辨，大多数网站都会考虑采用这种类似的开源框架验证码组件，省时省力同时效果也比较好。

在这次项目实践过程中，同时作为验证码生成者和验证码攻击者，经历矛与盾的较量觉得随着盾的不断变厚，也一定会出现更加尖锐的矛。即使随着文本验证码反分割反识别技术提高，相应的攻击者也会调整自己的策略来达到更高的识别率。不仅仅是简单的文本验证码，其他类型的如声音，滑动也是如此，会随着时间变化也会有越来越多的攻击方式能够成功破解。