网站投票防刷票验证码分析和实践

来源:[八方刷票公司](http://www.8fvote.com/) <http://www.8fvote.com>

**综述篇**

现如今，网站投票的方式被越来越多地用于各种评比活动中，互联网无疑为投票活动带来了极大的便捷，但是也由此产生了新的问题——刷票、或者准确的说是机器刷票。

回忆5、6年前各种选秀活动大量采用手机短信投票方式，从那时起刷票公司遍地开花，往后几年“圈钱”“骗钱”的手机短信方式逐渐减少，纯网站形式渐渐增多。也许就是从那时开始带热了刷票行业，到现在加上木马病毒灰色产业链的帮助，防刷票的技术门槛被大大的提高，刷票与防刷票已经演变为一场攻防较量。

想必所有正儿八经的投票主办方都不愿意自己的网络投票被人通过技术手段任意地增加票数，但是HTTP技术使然，想实现并采用比较有效的防刷票方式却又不是那么容易，或技术成本过高，或用户体验不佳。

本文正是希望通过分析，例举和对比各种验证方法，探索一些较好的防刷票验证方式，既能方便的实现和部署，又不失对用户的友好。

本文涉及的技术主要基于Linux、nginx、mysql、php-fastcgi构建的WEB平台（缩写为LNMP），包括gd库、curl库等，另外涉及的工具可能包括Firebug、Httpfox、Httpwatch、Fiddler、IE WebDeveloper。

文章将依次列出各种常见的验证码形式，对其为何被攻破的原因作出一定的阐述，可能由于个人水平有限，分析不到位或不准确，欢迎拍砖。

最后还得注明一点，本文谈的主要是防机器刷票，对于靠人肉战术，以拼人数，拼体力的方式刷票就没有什么太好的解决办法了。

**分析篇**

该部分将对基本而常见的验证方式进行分析。

**1.Cookie方式。**

这是一种很老的验证方式，原理是通过在浏览器上用cookie来记录相关信息，如最后一次投票的时间或已投票次数，如果用户再次投票时检测到超过了限制，则提示用户不能投票。

普通破解方式，可以通过浏览器的设置如IE的Internet选项，对cookie进行清空操作，而后cookie验证即宣告失败，一切计数从零开始。或者安装如Httpwatch、Fiddler、IE WebDeveloper等工具，这些工具中有按钮可以直接删除所有cookie或某些cookie。

程序破解方式，可以编写程序，对浏览器保存cookie的目录全部清空或部分删除。也可以使用如curl库，模拟浏览器访问，包括记录访问的cookie到某个文件中，而后在程序中清空或删除cookie文件即可。

有的网站虽然采用了cookie方式验证，但是却不验证浏览器到底支不支持cookie，到底从cookie中有没有取到数据，只是判断有cookie验证数据如何做，没有则跳过，这样的话，直接禁用cookie就实现了破解。

**2.IP方式**

IP限制曾经是比较有效的一种验证方式，由于IP的稀缺性决定了想要获取大量IP是存在一定难度的，因此IP限制一直到现在都被广泛使用。IP限制通过记录投票的IP地址、投票数量和时间来判断该IP用户是否超过投票限制，超过限制则不能投票。

普通破解方式，没有太简便的普通方式破解IP限制，如果有一定耐力的话，可以在PC上操作拨号连接，超过投票限制就断开重新拨号，这样可以获取一定数量的不同IP，但也不是每次都不同，毕竟电信运营商局端的IP池也是有限的，靠人工如此操作，效率很低人也很疲劳。

程序破解方式，可以使用一些拨号工具或代码，由程序来不断重复“拨号——投票——断开”的过程，但除了解放了人力外，能获取不同IP的数量仍然是比较有限的。有一些刷新软件中提供连接国内外代理服务器的方式，但是终究数量有限，而且访问国外代理服务器还存在连接困难，速度慢等问题。

还有一种程序破解方式，它的出现和使用让IP限制变的毫无作用。通过实际情况分析以及与[刷票公司](http://www.8fvote.com/)的沟通，推断和了解到刷票公司手中握有大量“肉鸡”，可以通过肉机进行投票操作，现在木马横行，肉机数量硕大，此种方式所能获取的IP可谓是海量的，IP限制面对此种方式完全失去了作用。

**3.表单验证串**

有的网页表单会以hidden形式保存一些随机的验证字符串，每次刷新表单这个值都会发生改变，提交表单时如果没有这个值或者值不符则提交失败。

要破解这种表单需使用如Httpfox或Httpwatch分析提交表单时的具体数据，结合Firebug或IE WebDeveloper或直接查看网页源码，便可找出这一随机字符串的变量名和数值。

编程时可以使用不同的方法（方法很多，不在此赘述）获取表单所在页面的源码，通过如正则表达式等方式从中截取出这个数值。

与此相同的道理，也可以把这个数值保存在session中，但只要刷票程序能按照HTTP协议规范模拟出正常的访问（如使用curl库），破解反倒更为容易。

总之，表单验证串方式还是很容易破解的，只要对HTML代码有一定了解，具备一定的编程能力，此种验证方式也仅能是稍稍增加了一点点破解的门槛而已。

**4.图片验证码**

图片验证码在包括投票在内的很多网页表单中都可以见到，相对来说，此种方式对破解者的编程能力要求比较高，因此门槛也高一些。但是，此等验证对“君子”足以，对“小人”还是收效不足。

常见的如数字验证码，字母验证码，数字和字母混合的验证码，现在有的网站还使用了中文的图片验证码。图片验证码具体还有多种样式，有使用纯色的背景或使用多色背景的，有带干扰像素或不带干扰像素。

早先开始的时候大都是纯色背景验证码，字母和数字之间的间隔固定不变，使用的颜色也相对单一，可能会带有一些噪点。对于这种形式的图片验证码，网上可以找到一些文章讲解如何破解，基本上是按固定宽度截图，所截字符图片与采样保存的字符图片进行像素比对识别，如果有噪点还需要去色去噪等。

而后，图片验证码被进一步改进，可能包含下述的一种或多种形式。如背景被设置得更加复杂，颜色更多，并且每次刷新表单时背景也在不断地变化；又如数字字母会随机变换字体，有粗有细，有大有小；还有就是可以让数字字母之间的间隔不固定，显示时对其角度有做出一定调整，使字符或左倾或右倾；再比如Discuz中那样，除了干扰曲线，还在背景中加入了浅色的干扰字符，值得借鉴。

以上形式在PHP中使用GD库均可实现，原以为如此这般定能达到很好的效果，结果却不是这样。如果采用早期对单色定宽的图片验证码所使用的破解方法，将费时费力，而实际情况则是基本很短时间就能被[刷票](http://www.8fvote.com/)者破解，因此推断有一些成熟的功能强大的OCR识别工具或dll库等被加以利用，可能还辅以一定的图片去色去噪的处理功能，这样，图片验证码的识别概率和效率都被大大提高。

当然，从实际的访问监控中发现，并不是每次刷新都能被正确识别，因此刷票者往往还采用“暴力”的方式对网站进行访问。即使尝试5次或10次才能成功一次，配合上对网站近乎疯狂地发送HTTP请求，最终也足以达其快速增加票数的目的了，反正在总请求数数量巨大的前提下，也能保证最终成功数不小。

还有一种形式的字母验证码，采用扭曲的方式来显示一组字母串，正如在谷歌google网站中经常见到的那种，对于防刷应该更加有效。但是，个人认为此种形式对广大的草根网民不够友好，可能很多人都无法正确地识别和输入，因此暂时就此一笔带过。

同样，使用中文的图片验证码现在也屡见不鲜，但是却要求用户能够进行中文输入，不管是用拼音也好，五笔也好，还是用手写板，终归是增大了难度，用户体验不佳。

最后附带说一下，网上有人说可以用MAC地址限制，这基本是不可行的，首先MAC地址可以修改，更重要的是，MAC地址只能在一个局域网中广播，不能跨过路由，又如何能在与用户之间被若干路由相隔的网站上轻易获取用户MAC。不过也许等到某天，政府建立了“大局域网”之后，没准还真就可以让MAC地址在整个网络中传播，以匹其“大局域网”之名。

**实践篇**

**1.投票处理程序须严谨**

投票处理程序应该先把好第一关，就是不给刷票者留下任何明显的漏洞。有经验的刷票者会由浅入深、从易到难地对可能存在的漏洞逐项进行侦测，有可被轻易攻击的漏洞存在，无疑可为刷票者带来极大地便利。

很多程序的编写者，往往只会按照业务或产品要求的主逻辑编写代码，功能是实现了，但是只要用户提交的请求略有不同或稍加修改，就可能绕到本不该进入的逻辑之中。也许有人要说这是测试人员的工作，但是网站能配备专职测试人员的怕是少之又少，因此，一方面既要实现主要逻辑，同时还要用缜密的心思周全考虑，查缺补漏完善代码，有一股子较真的精气神才不失为一名好的程序员。

下面从严谨编码的角度提一些个人意见和建议。

1）尽可能使用POST方式。常用的表单提交method有GET和POST方式，GET方式以明文参数带在链接后面传送，查看和识别相当容易，使用POST方式则需要借助专门工具才能查看到提交的内容。

2）检测是否支持Cookie。这个功能其实挺常见，如果关闭了Cookie，在访问一些论坛时会有相关提示，想必很多人都见过。检测Cookie的作用就是限制必须使用浏览器，或者编码HTTP请求和接收时需要处理头部的Cookie数据，又或者使用带Cookie支持的仿HTTP请求工具（如curl库）。

3）检测参数是否正确。可以检测提交的变量的名称是否正确，变量的个数是否相符，还可以进一步检测变量的类型是否符合要求，数值是否超出范围等等，一旦有不符之处则终止程序。

4）防范SQL注入。结合上一点来说，在对参数进行检测时，用相关函数处理变量，尤其是字符型变量，以防范SQL注入。虽说在投票处理程序中实用性可能不高，但是作为网络编程安全常常提起的重点，此处也不得不说一下。以PHP为例，常用的函数有addslashes()、 mysql\_real\_escape\_string()、mysql\_escape\_string()。

5）HTTP头部检测。可用来检测的HTTP头部信息主要有HTTP\_REFERER、HTTP\_USER\_AGENT、X-Forwarded-For。

HTTP\_REFERER信息表示用户请求当前页面的来源页面地址，就是说在来源页面操作之后才来到当前页面。可以对该信息的内容做检测，比如检测来源网址是否是本站地址，不是程序则做相应处理。

HTTP\_USER\_AGENT表示了用户所使用的客户端是什么，一般内容是各种浏览器的UA信息。可以检测当不是有效的浏览器数据时，程序做相应处理。

X-Forwarded-For包含了所谓的真实用户IP，意思是用户通过透明代理访问，代理服务器会把用户的真实IP保存在这个信息里，一般从WEB服务器日志中可以看到这个数据保存在每行日志的最后。

网上有不少获取用户IP的示例程序，先从X-Forwarded-For头中取值，没有再从REMOTE\_ADDR中取值，以确保获取最真实的用户IP，这个对网站来说可行，对投票来说并不可取，不应该在投票程序中使用X-Forwarded-For头获取用户IP地址。X-Forwarded-For是可以被模拟发送请求的，如果按此法取IP值，则投票程序的IP限制直接失效。个人建议，应该在投票程序中对包含X-Forwarded-For信息的HTTP请求采取诸如终止程序的措施。

6）表单重复刷新需处理。一般，WEB表单提交后点浏览器的后退按钮，会出现是否重新发送表单数据的提示框，如果投票程序未对此做处理，只需要后退刷新，再后退再刷新，则可实现刷票，相应的工具可以采用如屏幕按键精灵等来自动模拟鼠标点击。

7）最后一点，有if就一定有else。这一条要求编程者逻辑的严谨，意思是条件判断语句 if/elseif/else 要能完整覆盖同类条件下的不同状态或数值区间。

举个例子，曾今碰到过一个投票程序，通过浏览器投票时，进行了Cookie验证，当然每次清除Cookie可破，改用curl库请求，却发现无需开启Cookie支持即可直接提交投票请求。这说明，程序很可能只写了一个if来检测有Cookie数值时提示超过投票次数限制，而未写没有Cookie时该做什么，也就是说可能没有写相应的else或else中并没有加入诸如终止程序的逻辑。

上面几点仅是对编程的严谨提出建议，实际通过一步步地测试，最终大都可以模拟发送请求并通过程序的检测，因此，更大意义上做这些措施是对刷票多树立了一点门槛，兴许功夫不到家或是不耐烦的刷票者就被这几道小门槛挡在外面了呢。

还有一些具体到各种验证方式中的注意事项，留待后面对各种验证方式深入实践举例中详述。

**2.Cookie、IP地址等基本验证方式**

先看看Cookie是什么？Cookie最早是网景公司的前雇员Lou Montulli在1993年3月的发明。Cookie是由服务器端生成，发送给User-Agent（一般是浏览器），浏览器会将Cookie的key/value保存到某个目录下的文本文件内，下次请求同一网站时就发送该Cookie给服务器（前提是浏览器设置为启用cookie），Cookie名称和值可以由服务器端开发自己定义。

采用Cookie做验证是较早使用的一种表单验证方式，通过设置Cookie值记录投票次数和时间以达到限制目的。而Cookie自身的特点决定了使用此种方式的先天不足，极易破解，前文已经有所分析。

使用PHP或是javascript均可读写Cookie，为了和动态应用有更好的交互，建议使用PHP自带的方法。

PHP的写Cookie函数是setcookie，函数定义为：bool**setcookie** ( string $name [,string $value [, int $expire= 0 [,string $path [, string $domain [,bool $secure= false [,bool $httponly= false ]]]]]] )，而读Cookie数据则可以从$\_COOKIE[$name] 中直接获取。

再看看什么是IP地址。所谓IP地址就是给每个连接在Internet上的主机分配的一个32bit地址。按照TCP/IP协议规定，IP地址用二进制来表示，每个IP地址长32bit，比特换算成字节，就是4个字节。例如一个采用二进制形式的IP地址是“00001010000000000000000000000001”，这么长的地址，人们处理起来也太费劲了。为了方便人们的使用，IP地址经常被写成十进制的形式，中间使用符号“.”分开不同的字节。于是，上面的IP地址可以表示为“10.0.0.1”。IP地址的这种表示法叫做“点分十进制表示法”，这显然比1和0容易记忆得多。

可以这么理解，每个到访用户的IP地址是唯一的，因此可以把IP地址作为重要依据将用户和其投票行为绑定起来，也就是说，通过同时记录IP地址、投票时间和次数可以限制用户投票。很长一段时间以来，IP地址限制对投票验证起到了重要的作用，直到现在依然被广泛地使用，但是也存在问题，前文已经介绍过。

在上一点里提到过，投票程序不应从X-Forwarded-For头信息中获取IP，正确的应该是从REMOTE\_ADDR中取值，例如，$ip = $\_SERVER['REMOTE\_ADDR'] ，这样可以防止刷票者模拟X-Forwarded-For信息进行刷票。

**3.图片验证码**

先看看百度百科里对验证码的解释。验证码的英文CAPTCHA 这个词最早是在2002年由卡内基梅隆大学的Luis von Ahn、Manuel Blum、Nicholas J.Hopper以及IBM的John Langford所提出。CAPTCHA 是“Completely Automated Public Turing test to tell Computers and Humans Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机和人的公共全自动程序。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答CAPTCHA的问题，所以回答出问题的用户就可以被认为是人类。

从上面的解释得出结论，验证码的主要目的就是要规避非人类的投票行为，也就是说，可以认为识别输入了验证码且成功提交了表单的就是一个实实在在的人，而不是机器所为。使用验证码就是为了防止机器批量提交表单，辅以其他验证方式，能更有效杜绝刷票行为。

但事实是，随着机器识别能力的不断加强，普通的图片验证码已能够被比较容易的识破，同时双方也都在攻防过程中，不断地演变和进步，下面具体由简单到复杂地例举图片验证码（下文中有时简称为验证码）的各种形式。

1）**最原始的图片验证码**。由四位数字组成，也就是随机的一组数字字符串。最早时颜色单一，排列工整无任何歪斜，数字与数字之间间距像素固定不变，往后经过演变，数字颜色可能不再单一，可能是数字和字母随机出现并且位数被加长，也可能加入了一定量的干扰像素，但只要其仍然是整齐划一地排在一起且背景颜色单一，就注定还是最易破解的验证码。

图例：http://misc.clzg.cn/site/201003/31/39357_1270016925311G.jpghttp://misc.clzg.cn/site/201003/31/39357_1270016925c6GZ.jpg

2）**强干扰性的验证码**。上面中规中矩的验证码防刷功用几乎为零，逐渐出现了防刷效果更佳的验证码。验证码图片的大小变大了，变大的目的是为了给干扰元素更多的施展空间。

a.是用随机改变的背景图片。添加干扰像素点需要大量循环，对系统资源消耗较大，而准备若干背景图片，每次只需随机加载一张，便可获得有丰富的色彩变化的干扰效果。有的背景图片中还加入了浅色的数字或字母，更加大了干扰的作用。

b.字体随机改变。可以准备若干字体文件，用于生成时使用，好在英文字体文件普遍较小，对系统消耗不大。同时，字体颜色也随机生成，加大识别难度。但是比较正规的字体容易识别，可以掺杂一些机器难以识别的花体字，但全用稀奇古怪的字体也可能会给用户本身带来识别难度，需要注意。

c.字符随机倾斜和移位。可以取随机角度旋转字符，还可以在保证先后顺序可识别的前提下，让字符在上或下、左或右方向上随机移动，最终实现验证码东倒西歪、零零散散的效果。

图例：

http://misc.clzg.cn/site/201003/31/39357_1270027853vgJV.jpg（支付宝注册页面）

http://misc.clzg.cn/site/201003/31/39357_1270027853q87b.jpg（新浪博客留言）

**3.图片验证码举例分析**

1）**http://misc.clzg.cn/site/201003/31/39357_1270016925311G.jpghttp://misc.clzg.cn/site/201003/31/39357_1270016925c6GZ.jpg**

上图所示为最早出现、最简单、最基本的图片验证码。这种验证码，就是一个纯色背景加上数字或数字与字母的组合，避免被人从html源码中直接读取出数据。

此种验证码使用OCR工具识别非常容易，还可以用像素块直接比对识别。

2）http://misc.clzg.cn/site/201003/31/39357_1270027853vgJV.jpg

上图所示验证码加入了一定的背景像素干扰，但还是比较干净，数字的位置上下增加了错位显示，增加了一定识别难度，但总体来说破解仍然很容易。

3）http://misc.clzg.cn/site/201005/4/39357_1272938733ai41.jpghttp://misc.clzg.cn/site/201005/4/39357_1272938733Z232.jpg

上面的验证码在前两种的基础上有所改进，加入了较多的随机干扰像素，数字颜色变化比较丰富，每次刷新时包括数字的颜色和干扰像素的颜色都可能有所改变，而第二个相比第一个使用了比较特别的字体，并伴有一定倾斜，增加了识别难度。

4）http://misc.clzg.cn/site/201005/4/39357_1272938733SIdO.jpg

这个验证码下方有一条干扰曲线，使用大小写英文字母搭配，设置了较大的倾斜度和上下错位，这样captcha的宽高也就需要比较充裕的设置。

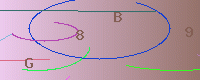
5）http://misc.clzg.cn/site/201005/4/39357_12729403279VK2.jpghttp://misc.clzg.cn/site/201005/4/39357_12729403274EE8.jpghttp://misc.clzg.cn/site/201005/4/39357_1272940327WNc5.jpg

上面这组验证码数字和字母混搭，使用了每次刷新随机变换的干扰背景，字体和颜色随机变换，并且伴有错位和倾斜，如果字体能选用一些更特别的花体可能效果更好，如下图。但是用太过特别的花体可能给草根用户带来较高人工识别的难度。

http://misc.clzg.cn/site/201005/4/39357_1272941139Wpig.jpg

6）http://misc.clzg.cn/site/201005/4/39357_12729387348H88.jpg

这个验证码相比其它加入了拖影效果，需要先对图片进行较大处理方能提高识别精度。

7）

这个验证码采用了gif格式，迷惑性较大，但倘若每次都固定最后一帧是真实验证码的话，使用工具取出gif的最后一帧再进行识别，难度也不是很高，这样看反倒浪费了系统资源来生成gif。

8）http://misc.clzg.cn/site/201005/4/39357_1272938734qjaQ.jpghttp://misc.clzg.cn/site/201005/4/39357_12729387343bhc.jpg http://misc.clzg.cn/site/201005/4/39357_12729416220mtm.jpg

这样的验证码国外使用的较多，机器识别难度较高。但是一方面对国人来说用户体验不佳，另一方面像PHP等动态网页应用中没有现成的扭曲函数可用，需要借助其他工具实现，移植性和通用性差。