CAPTCHAAs:

Are they really hopeless?

(Yes)
Completely Automated Public Test to tell Computers and Humans Apart

Why? Protect from automated abuse.
Examples

```
M9 QK J 9

No premium user. Please enter all letters having a 9 below.

LKOC138
```

```
kampfzig
```

```
Jmdig
```

```
Newtorn
WHITMAN
```
The Problem

- Really hard for a computer
- Hard for a person
- So they can’t be too hard
CAPTCHA Breaking

- Is it worth it?
- $$ value per CAPTCHA solved
- Retooling Cost
Automation?

- Is automation worthwhile?
- CAPTCHA “Farms”
- How much do they really cost?
- Accuracy requirement?
Two Approaches

- Attack the implementation
- Attack the actual CAPTCHA (by solving it)
Implementation Issues

- Not enough server side state
- Map request to token
- Has token been used before?

Problem:
- Try to break more than once
- Reuse solutions
function ts_is_human($ts_random, $string) {
    global $ts_random, $site_key;
    $datekey = date("F j");
    $rcode = hexdec(md5($site_key . $ts_random . $datekey));
    $code = substr($rcode, 2, 6);
    return $string==$code;
}

Source: http://coffelius.arabandalucia.com
Lousy Encoding

- CAPTCHA solution encoded in URL or form parameters

```
captcha_image.php?x=-8&y=20

<input type="hidden" name="cap" value="c4ca4238a0b923820dcc509a6f75849b"/>
```
Multiple Requests

- Different images for same CAPTCHA.
- Easy way to pump up accuracy.
Lousy RNG

- Most generators not secure
- Difficulty
  - Truncated output
  - Intermediate requests
  - Multiple servers
- Sleep easy. md5(rand)
Lousy RNG

Just for kicks, let's take a closer look.

```php
function generate_code($len) {
    $code = '';  
    for($i = 1; $i <= $len; $i++) {
        $code .= charset{rand(0, strlen(charset) - 1)};
    }
    return $code; 
}

Source: http://www.phpcaptcha.org
PHP rand

PHPAPI long php_rand(TSRMLS_D) {
    long ret;
    if (!BG(rand_is_seeded))
        php_srand(GENERATE_SEED() TSRMLS_CC);

    #ifdef ZTS
        ret = php_rand_r(&BG(rand_seed));
    #else
        #if defined(HAVE_RANDOM)
            ret = random();
        #elif defined(HAVE_LRAND48)
            ret = lrand48();
        #else
            ret = rand();
        #endif
    #endif
    return ret;
}
The `random()` function uses a non-linear additive feedback random number generator employing a default table of size 31 long integers to return successive pseudo-random numbers in the range from 0 to \((2^{31})-1\).

The random number generation technique is a linear feedback shift register approach, employing trinomials (since there are fewer terms to sum up that way).
For More...


Breaking CAPTCHA

• Recovering perl programmer ⇒ Lazy

• Off the shelf technology?
  • tesseract
  • jocr
  • ocrad
  • etc.
General Approach

- Use OCR engines as black box
- Additional pre / post processing to improve performance
Image Processing

- Remove / smooth noise
- Automate
  - PIL
  - ImageMagick
- Doesn’t add data, just makes more suitable for OCR engine.
Image Processing

Initial

Processed

$p^h_{cxb}$

$SLCAB$

$p^h_{cxb}$

$SLCAB$
Training Sets

- OCR engines need to be trained for particular fonts / styles
- Training can be automated
  - Use LaTeX, etc to generate data
  - Use actual CAPTCHA data
Other Strategies

• Attacking Audio CAPTCHAs
• Advantages?
  • Only one dimension
  • Frequency domain more intuitive
  • Less room for noise
Audio

- Time Domain
Guess where the numbers are?
Audio

- Frequency Domain
- Fourier Transform
  - Decompose (almost) any function in terms of complex exponentials
  - Perform frequency level filtering
  - More intuitive for sound than image
Power Spectral Density

[demos here]
What to do?

• Integrate cultural knowledge based on userbase.
• Hot or not CAPTCHA
• Cats vs. Dogs
Bourbon or Scotch?