Bab 8

Melawan Robot

Meskipun Internet saat ini tidak -atau belum?- menyerupai dunia a la Terminator seperti yang disugestikan oleh judul bab ini, tapi kehadiran robot-robot nakal yang gemar melakukan spamming dan bombing telah mengganggu kenyamanan. Terutama saat kita online sehingga telah menjadi masalah yang serius. Mulai dari form di situs web, mailserver, IM, handphone, bahkan log di server, dikotori oleh pesan-pesan spam. Dan ketidaksukaan seseorang terhadap orang lain dengan mudah berubah menjadi perbuatan saling menyerang dengan mailbombing, flooding, DDOS attack, dan lain sebagainya.

Spam atau bom dan serangan online merajalela karena mudah dan murah. Dan salah satu alasan utama mengapa mudah dan murah adalah karena bisa dilakukan secara otomatis oleh "robot"—program atau skrip yang berjalan otomatis tanpa pengoperasian manual oleh manusia. Jadi, untuk melawannya, kita pun perlu menempuh cara-cara otomatis.

Bab ini menjelaskan beberapa cara yang bisa ditempuh oleh skrip PHP dalam melawan robot yang melakukan spamming dan bombing.

Resep 8-1: Melawan Spambot dengan CAPTCHA

Spambot adalah skrip yang menjelajahi halaman web, mencari URL beberapa aplikasi forum, guestbook, atau blog yang popular dan mem-posting spam ke form komentar yang tersedia.

Cara yang ditempuh untuk melawan hal ini biasanya adalah sistem CAPTCHA (Completely Automatic Turing Test To Tell Computer and Human Apart atau uji otomatis untuk membedakan manusia dan mesin). Ada beberapa algoritma CAPTCHA yang telah dipakai orang. Misalnya menyuruh pengguna menebak kata yang tertera dalam sebuah gambar dengan tulisan yang terdistorsi. Atau menyuguhkan sebuah pertanyaan yang sulit dan jawabannya berupa kata atau frase. Misalnya,"Apa warna langit saat malam?" Jawabannya "hitam"). Atau menyuguhkan foto dan menyuruh pengguna menjawab beberapa pertanyaan. Seperti, "Berapa orang ada dalam foto ini?". Atau menperdengarkan sebuah file .WAV dan menyuruh pengguna menuliskan apa yang didengar.

Semua algoritma ini prinsipnya adalah menggunakan uji AI (Artificial Intelligence) yang saat ini masih sulit dipecahkan oleh komputer. Dengan kata lain, kita mencari kemampuan manusia yang sulit diduplikasi oleh komputer.

Manusia memiliki kemampuan mengenali tulisan meskipun tulisan tersebut terdistorsi cukup berat. Hal inilah yang dimanfaatkan sebagai salah satu uji CAPTCHA yang paling popular. Dengan melakukan uji ini, spambot dihadang karena tidak mampu menebak kata dalam gambar tulisan yang terdistorsi.

Resep di bawah merupakan sebuah implementasi CAPTCHA:

```
 1|<?
  |
 3|//
 4|// captcha.php
 5|//
  |
 7|$FONT_PATH = getcwd();
  |
 9|session_register('captcha_word');
  |
11|function make_captcha($word) {
```



```
12|  global $FONT_PATH;
  |
14|  // parameter untuk teks
15|  $angle = rand(-30,0);
16|  $size = rand(16,20);
17|  $fontfile = "$FONT_PATH/georgiaz.ttf";
18|  $coords = imagettfbbox($size, $angle, $fontfile, $word);
  |
20|  // ukuran image
21|  $width = abs($coords[4]-$coords[0])+60;
22|  $height = abs($coords[5]-$coords[1])+60;
  |
24|  // parameter distorsi
25|  $amplitude = rand(10,20);
26|  $period_factor = rand(1200,3600)/1000.0;
27|  $offset = rand(0,360);
  |
29|  // buat gambar
30|  $im = imagecreate($width, $height);
31|  $im2 = imagecreate($width, $height+$amplitude*2);
32|  $bg_color = imagecolorallocate($im,
33|                                 rand(192,255),
34|                                 rand(192,255),
35|                                 rand(192,255));
36|  $text_color = imagecolorallocate($im,
37|                                   rand(0,128),
38|                                   rand(0,128),
39|                                   rand(0,128));
40|  imagecopy($im2,$im,0,0,0,0,1,1);
41|  imagettftext($im, $size, $angle, $size, $size*1.5,
42|               $text_color, $fontfile, $word);
  |
44|  // distorsikan teks
45|  for ($i=0; $i<$width; $i++) {
46|    $y = sin(deg2rad($offset +
  $i*$period_factor))*$amplitude;
47|    imagecopy($im2, $im, $i, $y, $i, 0, 1, $height);
48|  }
  |
50|  return $im2;
51|}
```

```
   |
 53|if (!isset($show)) $show = '';
 54|if (!isset($captcha_word)) $captcha_word = '';
   |
 56|// user memasukkan tebakan katanya
 57|if (isset($word) && $captcha_word) {
   |
 59|  if (strtolower($word) == strtolower($captcha_word))
 60|    echo "Benar!";
 61|  else
 62|    echo "Salah!";
 63|  $captcha_word = "";
   |
 65|// tampilkan image
 66|} elseif ($show == 'image') {
   |
 68|  header("Content-Type: image/png");
 69|  imagepng(make_captcha($captcha_word));
   |
 71|// pilih kata dan tampilkan link ke image
 72|} else {
   |
 74|  // pilih kata dari kamus
 75|  if (($fp = @popen("grep '^.......\\?.\\?$' /usr/share/dict/
   words",
 76|                    "r"))!== false) {
 77|    $i = 0;
 78|    while ($line = fgets($fp, 8192)) {
 79|      if (rand(0, $i) == 0) $captcha_word = $line;
 80|      $i++;
 81|    }
 82|    $captcha_word = rtrim($captcha_word);
 83|  } else {
 84|    // pilih kata sebagai bilangan acak
 85|    $captcha_word = rand(10000000,99999999);
 86|  }
   |
 88|  echo "
   |
 90|    <h1>Captcha</h1>
   |
```



```
 92|    <p>Masukkan kata yang tertera di bawah ini. Jika kata
   kurang jelas,
 93|    tebaklah sedekat mungkin.
   |
 95|    <p><form method=POST>
 96|      <input name=word><input type=submit>
 97|      </form>
 98|    <p><img src=$PHP_SELF?show=image></p>
   |
100|  ";
   |
102|}
   |
104|?>
```

Pembahasan

Saat pengunjung pertama kali membuka halaman, maka akan dibawa ke baris 72-102. Kita mengambil sebuah kata acak dari file `/usr/share/dict/words`, yang defaultnya disertakan di distribusi Linux. Perhatikan perintah yang dikirimkan di popen() baris 72:

```
grep '^.......\?.\?$' /usr/share/dict/words
```

Rege pada grep di atas berarti kita hanya tertarik pada kata yang terdiri dari 6 hingga 8 huruf. Di komputer yang menggunakan Redhat 7.3, jumlah kata seperti ini ada 20.630 buah. Lalu, di baris 76-81 kita memilih satu kata acak di antara kata-kata tersebut. Algoritma yang digunakan di baris 76-81 ini telah dibahas di resep 2-4. Jika kita gagal membuka file `/usr/share/dict/words`, maka sebagai gantinya kita meng-generate bilangan acak untuk kata pilihan. Dibandingkan kata dalam kamus, bilangan acak kurang baik karena akan lebih tidak dikenali saat didistorsi. Kata dalam bahasa Inggris atau Indonesia yang dikenal pemakai akan bisa didistorsi lebih banyak karena otak pemakai biasanya sudah mengenali kata tersebut.
Setelah memperoleh kata pilihan, kita mengesetnya di variabel sesi `$captcha_word`. Kata ini akan digunakan di baris 66-69 untuk menampilkan gambar CAPTCHA.

Gambar CAPTCHA sendiri dihasilkan oleh fungsi `make_captcha()` di baris 11-51. Untuk membuat teks yang terdistorsi, pada resep ini saya menggunakan library GD dan mengandalkan fungsi `imagettftext()` dan `imagecopy()`. Pertama ukuran tulisan dengan ukuran font dan jenis font tertentu dicatat (baris 18), agar kita dapat memperkirakan ukuran kanvas gambar yang harus dibuat (baris 30-31). Semakin kompleks font, semakin baik karena akan semakin sulit dikenali oleh program. Biasanya

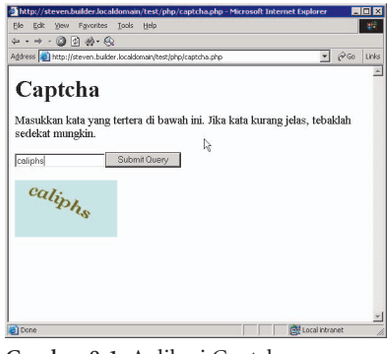
face italic juga dipilih karena kemiringan dan glyph-glyphnya lebih sulit dikenali. Di baris 31, dibuat sebuah image lain yang lebih besar tingginya. Ini karena nanti akan diterapkan gelombang sinus di baris 45-48 dari gambar berisi teks menjadi teks bergelombang. Caranya adalah mengkopi sekolom demi sekolom pixel dari kanvas pertama `$im` dan menggeser posisi vertikalnya di kanvas baru `$im2` menurut perhitungan `sin()` (baris 46 dan 47).

Selain cara ini, tentu Anda bisa menggunakan variasi lain dalam mendistorsi teks. Misalnya, menambahkan titik-titik di lokasi acak, seperti noise pada siaran televisi. Atau menambahkan gelombang sinus secara vertikal, agar gambar menjadi lebih 'lecek'. Atau menaruh latar belakang foto acak. Atau menggambar garis-garis dari berbagai arah, dan sebagainya. Sayangnya, memang library GD tidak seampuh **GIMP** atau **PhotoShop** dalam menerapkan efek-efek distorsi seperti ini.

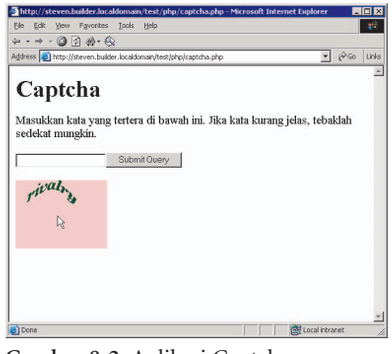
Baris 67-63 adalah bagian yang menerima submit form hasil tebakan pengunjung. Baik benar maupun salah, di baris 63 kita mereset `$captcha_word` agar sebuah gambar yang sama tidak bisa di-brute force dengan tebakan berulang-ulang. Karena ada sekitar 20.000 kata yang menjadi pilihan, maka sebuah spambot yang melakukan tebakan berulang-ulang hanya dapat berhasil dengan peluang 1 dari 20.000 kali submit. Ini praktis sudah membuat spambot tersebut menjadi tidak efektif. Jika ingin tebakan lebih sulit lagi, bisa juga ditambahkan beberapa digit random di akhir kata kamus.

Contoh resep ini beraksi bisa dilihat di Gambar 8-1 dan 8-2. Jika sebuah kata terlalu terdistorsi sehingga tidak bisa ditebak, pemakai dapat melakukan reload halaman untuk memperoleh kata baru.

Bagaimana menggunakan CAPTCHA untuk melindungi guestbook atau blog yang Anda buat? Pada form untuk men-submit komentar, Anda perlu menyertakan gambar CAPTCHA dan menyuruh pemakai menebak dan memasukkan kata dalam



Gambar 8-1. Aplikasi Captcha.



Gambar 8-2. Aplikasi Captcha.

gambar tersebut. Jika tebakan salah, Anda menolak submisi komentar. Memang, konsekuensi CAPTCHA adalah form tidak bisa dipakai oleh pengguna browser mode teks seperti **Lynx** atau **links**. Untuk pengguna browser-browser ini, bisa saja Anda menyuguhkan CAPTCHA varian lain. Misalnya dengan pertanyaan-pertanyaan teks. Yang perlu Anda ingat, pertanyaan yang disuguhkan tidaklah boleh mudah ditebak oleh program. Contoh yang dapat ditebak adalah:

1) Hanya ada 10 pertanyaan, sehingga program spambot bisa mengoleksi 10 pertanyaan berikut jawabannya.
2) Jawaban bisa dikalkulasi oleh program, misalnya: "Berapa 10 x 30?";
3) Jawaban selalu ada pada teks pertanyaan, misalnya: "Siapa di antara tokoh kartun berikut yang merupakan tikus: Mickey, Tom, atau Goofy?"

Untuk contoh-contoh tersebut, orang dapat dengan cukup mudah membuat spambot yang melakukan pencocokan string, operasi regex, atau mengambil kata dari pertanyaan dan melakukan brute-force.

Di resep 10-3, kita akan menerapkan CAPTCHA ini dalam membuat guestbook.

Resep 8-2: Melawan Leecher

Yang disebut leecher biasanya adalah program atau bisa juga orang yang melakukan download secara masal. Di dunia P2P, biasanya leecher berarti orang yang inginnya men-download saja tapi tidak mau upload atau sharing. Leecher menyebalkan dan juga merugikan.

Pada konteks situs web, leecher biasanya berupa **program downloader** yang ingin men-download sebuah file tanpa mengunjungi halaman web kita. Padahal kita ingin agar orang mengunjungi dulu halaman download agar dapat melihat banner. Leecher juga bisa berarti situs lain yang memiliki link ke file-file milik kita (atau disebut **outside links**). Dengan kata lain, situs-situs tersebut menyuguhkan file di situs mereka tapi tidak mau meng-host file-nya, melainkan 'nebeng' ke situs kita.

Bagaimana mencegah program downloader dan outside links seperti ini? Ada beberapa cara. Yang pertama adalah dengan mengecek header HTTP `Referer` (`$_ENV['HTTP_REFERER']`). Kita dapat memproteksi halaman download dengan mewajibkan `HTTP_REFERER` mengandung nama situs kita pula:

```
if (!isset($_ENV['HTTP_REFERER']) ||
    !preg_match("#^http://(www\.)?steven\.localdomain/#i",
                $_ENV['HTTP_REFERER'])) {
  die("Outside links tidak diperbolehkan!");
```

```
  // die() di atas bisa juga diganti dengan header("HTTP/1.0 403
Forbidden");
  // diikuti exit(). tapi jika mengirimkan respon Forbidden,
pastikan dokumen
  // anda tidak akan dicache sebab pengguna valid bisa saja
terkena cache
  // forbidden
}

// ...
```

Dengan kata lain, halaman download harus diikuti dari halaman lain di situs kita juga. Tidak bisa dari halaman situs lain atau diketikkan langsung URL-nya dari address bar. Sayangnya, header `Referer` mudah ditipu atau digenerate oleh program seperti **Wget**. Misalnya, perintah wget di bawah ini mengambil sebuah halaman http://steven.localdomain/dua.php dengan berpura-pura berasal dari halaman lain http://steven.localdomain/satu.php:

```
$ wget –header "Referer: http://steven.localdomain/satu.php" \
    http://steven.localdomain/dua.php
```

Cara kedua adalah dengan mengecek header `User-Agent`. Setiap browser mengirim header User-Agent yang unik. Misalnya IE 6.0 di Win2k mengirim:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;
Q312461; .NET CLR 1.0.3705)"
```

Sementara Mozilla 1.5b mengirimkan:

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US;
rv:1.5b)
Gecko/20030827
```

Program seperti **Wget** atau **GetRight** pun default-nya mengirimkan `User-Agent` masing-masing. Kode berikut dapat dipakai untuk hanya mengizinkan IE, Mozilla, dan Opera:

```
if (!isset($_ENV['HTTP_USER_AGENT']) || (
    !preg_match("#Mozilla/.+rv:.+Gecko/\d+#") && # mozilla, ns6/7
& firebird
    !preg_match("#MSIE \d+#") &&                # msie
```



```
    !preg_match("#Opera[/ ]#")                  # opera
    )) {
  die("Hanya pergunakan IE, Mozilla, atau Opera!");
  // die() di atas bisa juga diganti dengan header("HTTP/1.0 403
Forbidden");
  // diikuti exit(). tapi jika mengirimkan respon Forbidden,
pastikan dokumen
  // anda tidak akan dicache sebab pengguna valid bisa saja
terkena cache
  // forbidden
}

// ...
```

Sayangnya lagi, header `User-Agent` pun sama saja mudah dipalsukan. Misalnya dengan wget:

```
$ wget -U "MSIE 7.0" http://steven.localdomain/tiga.php
```

Cara ketiga dengan cookie. Anda bisa mewajibkan pengguna login dulu ke situs , atau menerima cookie dulu . Dengan keanggotaan, praktis masalah seperti outside links akan terselesaikan, tapi akan menyebabkan kerepotan bagi pemakai Anda. Dengan memberi cookie di sebuah halaman tertentu, misalnya halaman download, Anda bisa memastikan bahwa halaman tersebut dikunjungi dulu.

Cara keempat adalah dengan menyembunyikan link `<a href=>` menjadi link Javascript. Ini cukup ampuh dalam melumpuhkan Wget dan program-program sejenis. Contoh, kita bisa mengubah:

```
<a href=file-1.2.4.tar.gz>Download</a>
```

menjadi:

```
<a href=# onClick="dl(5)">Download</a>
```

di mana `dl()` adalah fungsi Javascript yang didefinisikan di bagian sebelumnya. Misalnya:

```
<script>
function dl(nomor) {
  url="";
```

```
  if      (nomor==1) url="file-1.2.3.tgz";
  else if (nomor==2) url="file-1.2.4.tgz";
  else if (nomor==3) url="file-1.2.5.tgz";
  else if (nomor==4) url="file-1.2.6.tgz";
  else if (nomor==5) url="file-1.2.7.tgz";

  document.location.href=url;
}
</script>
```

Resep di bawah ini menggunakan gabungan cara-cara di atas untuk mempersulit leecher men-download menggunakan downloader atau outside links.

```
 1|<?
  |
 3|//
 4|// download-menu.php
 5|//
  |
 7|$FILES_DIR = "./fonts";
  |
 9|@chdir($FILES_DIR) or die("Can't chdir!\n");
10|$files = array();
11|if ($dir = @opendir(".")) {
12|  while (($file = readdir($dir)) !== false) {
13|    if (is_file($file) && $file != ".htaccess") $files[] =
   $file;
14|  }
15|}
  |
17|setcookie("visit_menu", 1);
  |
19|// generate JS
20|echo "<script>\n";
21|echo "function dl(nomor) {\n";
22|echo "  url=\"\";\n";
23|for ($i=0; $i<count($files); $i++) {
24|  echo "  ",
25|       ($i==0 ? "if " : "else if "),
26|       "(nomor==$i) url=\"$files[$i]\";\n";
27|}
```



```
28|echo "  document.location.href = \"download.php/\" + url;\n";
29|echo "}\n";
30|echo "</script>\n";
  |
32|echo "<h1>Download font gratis!</h1>";
33|echo "<ul>";
34|for ($i=0; $i<count($files); $i++) {
35|  echo "<li><a href=# onClick=\"dl($i)\">$files[$i]</a> ".
36|       "(".filesize($files[$i])." bytes)";
37|}
38|echo "</ul>";
  |
40|?>
```

```
 1|<?
  |
 3|//
 4|// download.php
 5|//
  |
 7|$FILES_DIR = "./fonts";
 8|$SITE = "steven.builder.localdomain";
  |
10|# saring karakter berbahaya, terutama '/' atau '\'
11|if (!isset($PATH_INFO)) $PATH_INFO = "/";
12|$file = preg_replace("/[^A-Za-z0-9_. -]+/", "", $PATH_INFO);
  |
14|$path = "$FILES_DIR/$file";
  |
16|if (!$file || !file_exists($path)) {
17|  header("HTTP/1.1 404 Not Found");
18|  echo "<h1>404 Tidak Ditemukan!</h1>";
19|  exit;
20|}
  |
22|header("Cache-Control: no-cache");
23|header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
  |
25|if (!isset($_COOKIE['visit_menu'])) {
26|  header("Location: ../download-menu.php");
27|  exit;
```

```
28|}
  |
30|if (!isset($_ENV['HTTP_REFERER']) ||
31|    !preg_match("#^https?://$SITE/#", $_ENV['HTTP_REFERER'])) {
32|  header("Location: ../download-menu.php");
33|  exit;
34|}
  |
36|$size = filesize($path);
37|$lastmod = "Last-Modified: ". gmdate("D, d M Y H:i:s",
38|                                    filemtime($path)) . "
   GMT";
  |
40|if (!($fp = fopen($path, "r"))) {
41|  header("HTTP/1.1 500 Internal Server Error");
42|  echo "<h1>505 Tidak Bisa Didownload!</h1>";
43|  exit;
44|}
  |
46|header("Content-Type:  application/octet-stream");
47|header($lastmod);
48|header("Content-Length:  $size");
  |
50|$left = $size;
51|while ($left > 0) {
52|  $bytes = $left > 8192 ? 8192 : $left;
53|  echo fread($fp, $bytes);
54|  $left -= $bytes;
55|}
  |
57|fclose($fp);
  |
59|?>
```

Resep terdiri dari dua file, `download-menu.php` dan `download.php`. download-menu.php membaca daftar file dan membuat daftar link beserta kode Javascript yang dibutuhkan. `download.php` adalah skrip yang membungkus atau menjaga download. Direktori di `$FILES_DIR` harus dilindungi dengan `.htaccess` atau ditaruh di luar akses webserver agar leecher tidak dapat mem-bypass `download.php` dengan langsung men-download dari file-file di direktori tersebut. Resep ini dapat dikombinasikan dengan resep 3-3 agar download dapat di-resume.

Resep 8-3: Melawan Downloader: Black Hole

Seringkali jika ada orang ingin men-download semua halaman di sebuah situs sekaligus. Mereka menggunakan program downloader yang dapat mengikuti link dari homepage terus hingga semua pojok-pojok situs tertelusuri. Jika program downloader bekerja tanpa delay bahkan paralel, dapat menyebabkan situs terbebani. Salah satu cara menghukum downloader adalah dengan membuat jebakan atau *black hole* seperti ini:

```
 1|<?
  |
 3|//
 4|// blackhole.php
 5|//
  |
 7|$prefix = isset($PATH_INFO) ? "" : "blackhole.php/";
  |
 9|for($i=1; $i<=5; $i++) {
10|  echo "<a href=$prefix".rand(1,1000)."/>link $i</a><br>\n";
11|}
  |
13|?>
```

Kita lalu memasang link ke black hole ini di halaman lain:

```
<a style="text-decoration: none" href=blackhole.php/> </a>
```

Karena link ini tidak terlihat oleh manusia, maka pengunjung manusia tidak akan mengkliknya. Tapi downloader atau crawler yang mengikuti link akan mem-parse dan mengikuti link ini.

`blackhole.php` akan mengembalikan beberapa link relatif ke dirinya sendiri, namun dengan `PATH_INFO` yang semakin dalam levelnya. Misalnya, jika pertama kali dipanggil dengan `blackhole.php/`, hasilnya adalah:

```
<a href=322/>link 1</a><br>
<a href=796/>link 2</a><br>
<a href=575/>link 3</a><br>
<a href=197/>link 4</a><br>
<a href=221/>link 5</a><br>
```

Downloader akan mengikuti link ini. Link pertama menjadi blackhole.php/322/. Ketika diikuti, `blackhole.php` akan terpanggil kembali, kali ini dengan `$PATH_INFO "/322/"`. Skrip akan mengembalikan:

```
<a href=678/>link 1</a><br>
<a href=6/>link 2</a><br>
<a href=182/>link 3</a><br>
<a href=678/>link 4</a><br>
<a href=549/>link 5</a><br>
```

Download akan mengikuti `blackhole.php/322/678/`, dan seterusnya. Inilah sebabnya mengapa disebut sebagai black hole, karena jika diikuti terus, downloader tidak akan pernah keluar dari `blackhole.php`, melainkan akan terus terjerumus semakin dalam. Hingga biasanya akhirnya harddisk si pendownload menjadi penuh oleh direktori-direktori kosong. Atau program downloader crash atau menyerah karena struktur direktori terlalu dalam.

Resep 8-4: Melawan Downloader: Speed Cop

Kadang-kadang kita tidak keberatan ada downloader yang menyedot isi situs, selama downloader tersebut "sopan" atau tidak men-download terlalu cepat. Kita dapat membatasi kecepatan download dalam jumlah halaman/detik sebuah klien dengan cara seperti ini:

```
 1|<?
  |
 3|//
 4|// speed-cop.php
 5|//
  |
 7|$DB_HOST = "localhost";
 8|$DB_DB = "speedcop";
 9|$DB_USER = "speedcop";
10|$DB_PASS = "toofast!";
11|$MEMORY_PERIOD = 3600;   # pengukuran kecepatan dalam rentang
   1 jam
12|$LIMIT = 250;            # maksimum 250 halaman dalam 1 jam
13|$BLOCK_PERIOD = 24*3600; # pemblokiran selama 1 hari
```



```
  |
15|$db = @mysql_connect($DB_HOST, $DB_USER, $DB_PASS)
16|  or die("Can't connect to MySQL");
17|mysql_select_db($DB_DB);
  |
19|$now = time();
20|$res = mysql_query("select * from hits where
   ip='$_ENV[REMOTE_ADDR]'");
21|$row = mysql_fetch_array($res);
22|if (!$row) $row = array('ip' => $_ENV['REMOTE_ADDR'],
23|                        'start_time' => $now,
24|                        'block_time' => 0,
25|                        'is_blocked' => 0,
26|                        'hits' => 0,
27|                        '_new' => 1,
28|                        );
29|if ($row['is_blocked']) {
  |
31|  if ($row['block_time'] < $now-$BLOCK_PERIOD) {
32|    // saatnya blokir dibuka
33|    $row['is_blocked'] = 0;
34|    mysql_query("update hits set is_blocked=0, hits=1,
   start_time=$now
35|                  where ip='$_ENV[REMOTE_ADDR]'");
36|  }
  |
38|} else {
  |
40|  if ($row['start_time'] < $now-$MEMORY_PERIOD) $row['hits']
   = 0;
41|  $row['hits']++;
42|  if ($row['hits'] > $LIMIT) {
43|    $row['is_blocked'] = 1;
44|    mysql_query("update hits set is_blocked=1,
   block_time=$now
45|                  where ip='$_ENV[REMOTE_ADDR]'");
46|  } else {
47|    if (isset($row['_new'])) {
48|      mysql_query("insert into hits (ip,start_time,hits)
   values
49|                    ('$_ENV[REMOTE_ADDR]',$now,$row[hits])");
```

```
 50|    } else {
 51|      mysql_query("update hits set hits=$row[hits],
 52|                      start_time=$row[start_time]
 53|                    where ip='$_ENV[REMOTE_ADDR]'");
 54|    }
 55|  }
 56|}
   |
 58|if ($row['is_blocked']) die("Blocked!");
   |
 60|echo "Normal content...";
   |
 62|?>
```

Pembahasan

Sebelum kode berjalan siapkan database dan tabel MySQL dengan struktur:

```
create table hits (
  ip varchar(15) not null primary key,
  start_time int unsigned not null,
  block_time int unsigned not null,
  is_blocked tinyint unsigned not null,
  hits int not null
);
```

Baris 7-58 perlu ada di setiap halaman situs yang ingin diproteksi. Bagian pengecekan bekerja dengan mencari record di tabel `hits` untuk mengetahui berapa jumlah hit yang dibuat klien dalam selang waktu tertentu. Jika klien ini baru mengunjungi halaman lagi setelah sekian lama (lebih dari `$MEMORY_PERIOD`) maka kita melupakan hit yang lama dan mengesetnya menjadi 1 kembali (baris 40-41).

Jika jumlah hit telah melebih limit `$LIMIT` (baris 42) maka kita mengeset field `is_blocked` menjadi 1 sebagai tanda bahwa klien ini diblok sementara karena mendownload terlalu cepat. Sementara jika belum mencapai limit, kita mengupdate record di database untuk menaikkan counter hit.

Blokir baru dibuka setelah `$BLOCK_PERIOD` berlalu (di resep diset 24 jam). Kode untuk melakukan ini di 31-36. Setelah blokir dibuka, counter hit kembali direset menjadi 1.

Catatan: kita melakukan pengeblokan berdasarkan IP dan bukan sesi/cookie, karena sesi/cookie dapat diabaikan oleh klien.