11. Kaufmann, A. and Gupta, M.M., 1988, Fuzzy Mathematical Models in Engineering and Management Science, North-Holland, Amsterdam.

12. Kaufmann, A. and Gupta, M.M., 1991, Introduction to Fuzzy Arithmetic Theory and Applications, Van Nostrand Reinhold, New York.

13. Ke, T., 1992, "Target decision by entropy weight and fuzzy," System Engineering Theory and Practice, Vol. 5 (in Chinese).

14. Mon, D.L., Cheng, C.H. and Lin, J.C., 1994, "Evaluating weapon system using fuzzy analytic hierarchy process based on entropy weight," Fuzzy Sets and Systems, Vol. 62, pp. 127-134.

15. Saaty, T.L., 1980, The Analytical Hierarchy Process, McGraw Hill, New York.

16. Yeh M-F., Lu H-C. Evaluating Weapon Systems Based on Grey Relational Analysis and Fuzzy Arithmetic Operations / Journal of the Chinese Institute of Engineers, Vol. 23, No. 2, 2000, pp. 211-221.

17. Zadeh, L.A., 1965, "Fuzzy sets", Information and Control, Vol. 8, pp. 338-353.

18. Zimmermann, H.J., 1991, Fuzzy Set Theory - and Its Applications, second, revised edition, Kluwer Academic Publishers, Boston.

УДК 004.921

ДОСВІД РЕАЛІЗАЦІЇ CAPTCHA З ВИКОРИСТАННЯМ ЕЛЕМЕНТУ HTML5 CANVAS

Ю.С. Носач[1], Г.М. Кодола[2], Н.С. Волинець[3]

[1]студент, Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», м. Дніпропетровськ, e-mail: y-nosach97@yandex.ua

[2]викладач кафедри інформаційних систем, Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», м. Дніпропетровськ, e-mail: gkodola@gmail.com

[3]викладач кафедри інформаційних систем, Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», м. Дніпропетровськ, e-mail: nm80@mail.ru

Анотація. У роботі викладений аналіз основних правил формування CAPTCHA та алгоритмів перетворень зображень з білінійною фільтрацією. Описаний досвід реалізації CAPTCHA з використанням елементу HTML5 Canvas.

***Ключові слова**: CAPTCHA, HTML5, CANVAS, піксельні перетворення.*

CAPTCHA DEVELOPMENT EXPERIENCE USING ELEMENT OF HTML5 CANVAS

Yurii Nosach[1], Galyna Kodola[2], Natalya Volynec[3]

[1] Student of the State Higher Educational Institution "Ukrainian State Chemical Technology University", Dnepropetrovsk, e-mail: y-nosach97@yandex.ua

[2]Lecturer Department of Information Systems of the State Higher Educational Institution "Ukrainian State Chemical Technology University", Dnepropetrovsk, e-mail: gkodola@gmail.com

[3]Lecturer Department of Information Systems of the State Higher Educational Institution "Ukrainian State Chemical Technology University", Dnepropetrovsk, e-mail: nm80@mail.ru

Abstract. This paper presents an analysis of the basic rules of forming CAPTCHA algorithms and transformations of images with bilinear filtering. The experience of implementing CAPTCHA element using HTML5 Canvas described.

***Keywords**: CAPTCHA, HTML5, CANVAS, pixel transformation.*

Вступ. Як і десятиріччя тому, проблема боротьби зі спамом (англ. spam – масова розсилка кореспонденції рекламного чи іншого характеру людям, які не висловили бажання її одержувати) залишається сьогодні надзвичайно актуальною. Увесь спам можна розділити на ручний та автоматичний. Якщо проблема ручного спаму дуже лего вирішується шляхом обов'язкової модерації усіх вхідних повідомлень перед їх публикацією, то автоматичний спам проконтролювати набагато важче через його набагато більший об'єм. Часто на різні сайти та форуми потрапляють рекламні роботи, які переповнюють сторінки своїми «спамерськими» повідомленнями. Користувачам стає все складніше розібратися у великій кількості непотрібної інформації. Також ці повідомення викликають додаткове навантаження на серевер, чим ще більше ускладнюють доступ до сайтів.

Ціль роботи. Адміністратори сайтів все частіше стикаються з такими проблемами:

- сторінки і форуми, які «тонуть» під вагою спаму;
- облікові записи, які створені під помилковим приводом для неправомірних цілей;
- боти, які руйнують динаміку сайту;
- необхідність постійного контролю за якістю контенту і досвідом користувачів.

Тому постає проблема, як не допустити виконання скрипта на веб-сторінці без участі людини.

Матеріали і результати досліджень. Впродовж останнього десятиріччя внаслідок широкої діяльності спам-ботів здобув великої популярності метод захисту веб-сторінок з використанням так званої «капчі» (англ. *captcha – completely automated public turing test to tell computers and humans apart* – повністю автоматизований публічний тест Тюринга для розрізнення комп'ютерів і людей). Цей термін з'явився 2000-го року в університеті Карнегі-Меллона у штаті Пенсільванія, США [1].

Внаслідок широкої діяльності спам-ботів було винайдено декілька способів захисту веб-сторінок:

1. reCAPTCHA від Google. Користувачу пропонується поставити відмітку в певній області форми, для того, що засвідчити, що він – людина.

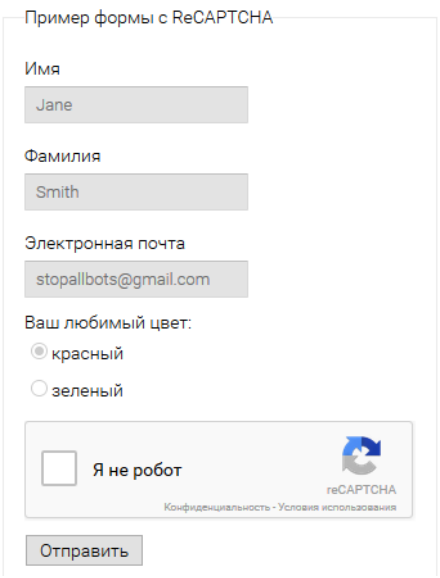

Рисунок 1 – reCAPTCHA від Google

2. CAPTCHA, де потрібно обрати зайвий варіант:

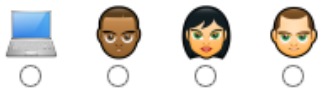

Рисунок 2 – CAPTCHA, де потрібно обрати зайвий варіант

3. CAPTCHA, де потрібно обрати вірне зображення:

Рисунок 3 – CAPTCHA, де потрібно обрати вірне зображення

4. CAPTCHA, де потрібно розв'язати математичний приклад:

Рисунок 4 – CAPTCHA, де потрібно розв'язати математичний приклад

5. CAPTCHA, де за допомогою повзунків потрібно вирівняти зображення:

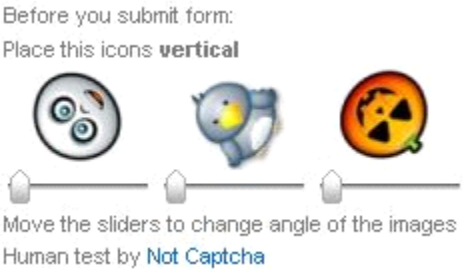

Рисунок 5 – CAPTCHA, де за допомогою повзунків потрібно вирівняти зображення

У найпоширенішому варіанті капча генерується зображення з послідовністю довільних символів та додаванням напівпрозорості, затінення, а також різного виду шумів та перетворень. Рідше застосовуються капча, засновані на розпізнаванні мови (в основному як альтернатива для людей з порушеннями зору), або на інших варіантах завдань штучного інтелекту. Це такий собі тест, який використовується для того, щоб визначити хто використовує систему – людина чи комп'ютер. Фактично, основною метою капчі є недопущення виконання скрипта без участі людини.

Основні властивості капчі:

- стійкість до розпізнавання – властивість, яка захищає капчу від автоматичного розпізнавання за допомогою безлічі відповідних сервісів;
- стійкість до відгадування – властивість, що не дозволяє відгадати її значення за невелику кількість спроб. Якщо набір можливих значень капчі невеликий, то підібрати потрібний варіант автоматично буде дуже просто.

Основні правила проектування капчи [2]:

1. Шуми – перш за все, повинні бути присутніми шуми. Тут краще за все використовути різнокольорові лінії, що в довільних місцях перетинають зображення (рис. 6):

Рисунок 6 – Застосування шумів

2. Шрифт – досить надійним способом захисту від розпізнавання буде використання найбільш шрифтів з курсивом та різного виду засічками, але

головне не перестаратися. Також можна для кожного символу використовувати свій шрифт (рис. 7):

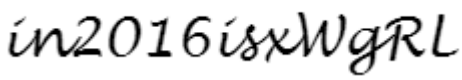

Рисунок 7 – Використання різноманітних шрифтів

3. Кольори – зображення повинно містити хоча б декілька різних кольорів. Дуже доречно буде застосувати як для фону, так і для літер лінійний або радіальний градієнт з використанням від 3 до 5 довільних кольорів (рис. 8):

Рисунок 8 – Застосування градієнту

4. Відстань між літерами – чим меншою вона буде, тим більше перешкод для автоматичного розпізнавання, оскільки при малій відстані літери будуть мало не злипатися. Але тут головне не перестаратися, бо занадто велике зближення символів призведе до значного погіршення розпізнавання людиною (рис. 9):

Рисунок 9 – Зміна міжсимвольного інтервалу

5. Розміри символів – досить дієвим способом захисту від ботів буде використання різних розмірів для кожного символу. Це становитиме досить значну перешкоду автоматичному розпізнаванню (рис. 10):

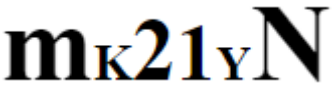

Рисунок 10 – Зміна розміру шрифту для кожного символу

6. Нахилення символів – треба обрати певний діапазон кута, на який буде нахилено кожен символ. Зловживати цим способом не варто, бо це призведе до того, що один символ буде «напливати» на інший (рис. 11):

Рисунок 11 – Зміна нахилу символів

7. Використання динамічних перетворень – мабуть, найдієвіший спосіб захисту від ботів (рис. 12):

Рисунок 12 – Застосування динамічних перетворень

Методи динамічних перетворень полягають у застосуванні до зображення піксельних перетворень з білінійною фільтрацією. Внаслідок використання цього методу зображення може бути трохи стиснуте або розтягнуте з «плаваючими» символами. Вибір цих перетворень досить великий і обмежений лише нашою фантазією та математикою. Тож розглянемо лише деякі з них.

Найчастіше використовуються три способи перетворень: «Twirl», «Spherize» та «Pyramid» [3]. Їх зручність полягає в тому, що вони майже завжди перетворюють зображення, не погіршуючи читабельність тексту з боку користувача, і одночасно ускладнюють автоматичне розпізнавання.

Всі перетворення, які розглядаються, реалізовані наступним чином: у функцію на вхід надходять координати отримуваного зображення, а результатом виконання – координати вихідного зображення.

1. Метод *«Twirl»:*

Вхід: (x_0, y_0) – координати отримуваного зображення

Вихід: (x, y) – координати вихідного зображення

width – ширина зображення

height – висота зображення

Розрахунок нових координат вихідного зображення проводиться за формулами (1-4):

$$x = x_0 - \frac{width}{2} \quad (1)$$

$$y = y_0 - \frac{height}{2} \quad (2)$$

$$r = \sqrt{x^2 + y^2} \quad (3)$$

$$maxr = \frac{width}{2} \quad (4)$$

Якщо $r > maxr$, то повертаємо вхідні значення (x_0, y_0), інакше розраховуємо за формулами (5-7):

$$a = arctg(y, x) + 1 - \frac{r}{maxr} \quad (5)$$

$$dx = cos(a) * r \quad (6)$$

$$dy = sin(a) * r \quad (7)$$

Повертаємо: ($dx + \frac{width}{2}$; $dy + \frac{height}{2}$)

2. Метод *«Spherize»:*

Вхід: (x_0, y_0) – координати отримуваного зображення

Вихід: (x, y) – координати вихідного зображення

width – ширина зображення

height – висота зображення

Розрахунок нових координат вихідного зображення проводиться за формулами (8-11):

$$x = x_0 - \frac{width}{2} \quad (8)$$

$$y = y_0 - \frac{height}{2} \quad (9)$$

$$r = \sqrt{x^2 + y^2} \quad (10)$$

$$maxr = \frac{width}{2} \quad (11)$$

Якщо $r > maxr$, то повертаємо вхідні значення (x_0, y_0), інакше проводимо розрахунок нових координат за формулами (12-15):

$$a = arctg(y, x) + 1 - \frac{r}{maxr} \quad (12)$$

$$k = \left(\frac{r}{maax}\right)^2 * 0{,}5 + 0{,}5 \quad (13)$$

$$dx = cos(a) * r * k \quad (14)$$

$$dy = sin(a) * r * k \quad (15)$$

Повертаємо: ($dx + \frac{width}{2}$; $dy + \frac{height}{2}$)

3. Метод *«Pyramid»:*

Вхід: (x_0, y_0) – координати отримуваного зображення

Вихід: (x, y) – координати вихідного зображення

width – ширина зображення

height – висота зображення

Розрахунок нових координат вихідного зображення проводиться за формулами (16-20):

$$x_0 = x_0 - \frac{height}{2} \quad (16)$$

$$y_0 = y_0 - \frac{height}{2} \quad (17)$$

Визначаємо:

$$dist = \max(|x_0|, |y_0|) \quad (18)$$

$$dx = x_0 * \frac{dist}{width} * 2 \quad (19)$$

$$dy = y_0 * \frac{dist}{height} * 2 \quad (20)$$

Повертаємо: $(dx + \frac{width}{2}; dy + \frac{height}{2})$

При реалізації капчі обов'язково треба врахувати деякі технічні нюанси:

1. Значення для перевірки (тобто рядок, який треба ввести) повинно зберігатися на сервері, а не передаватися разом із зображенням. Для зіставлення відвідувача і правильного значення капчі необхідно використовувати якийсь ключ, який передається разом з капчею (ідентифікатор сесії, номер капчі і т.п.).

2. Захист від перебору. Якщо ваша капча стійка до розпізнавання, але не дуже стійка до перебору (наприклад, складається з декількох цифр), бажано обмежити число неправильних відповідей з одного ip/одного логіну.

3. Захист від DoS атак. Слід розуміти, що генерація капчі на своєму сервері – це зручний спосіб для DoS атак з боку хакерів. Тож слід обмежити число генерацій капчі для одного IP.

Створюючи надійну від роботів капчу, не варто також і забувати про зручність використання з боку звичайного користувача, тож сформулюємо деякі правила, щоб надійна від роботів капча не становила значних перешкод для користувача:

1. Якщо в наборі символів використовується кириличний алфавіт, обов'язково треба виключити літеру «ё». Це пов'язано з тим, що на багатьох комп'ютерах ця клавіша налаштована на якусь програмну дію. І взагалі, краще використовувати лише латинські символи та цифри.

2. Не варто використовувати одночасно великі та малі літери. Внаслідок динамічних перетворень розміри символів на зображенні можуть змінитися і користувачу буде важко зрозуміти, мала то літера чи велика.

3. Слід уникати схожих символів. Насамперед це стосується цифри 0 та літер «O», а також цифри 1 та літери «I». Знову ж таки, внаслідок використання перетворень буде неможливо зрозуміти що то за символ.

4. І, мабуть, найголовніше правило – це уникати використання капчі там, де без неї можна обійтися. Тим більше у випадках, коли й так зрозуміло, що дія виконується саме користувачем, а не роботом.

Зловживання капчею на сайті лише знизить зручність користування сайтом, а також значно підвищить навантаження на сервер.

В даній роботі було виконано створення капчі за допомогою елементу canvas, на якому генерується текст для перевірки.

Для ускладнити автоматичного розпізнавання тексту, було відібрано найнерозбірливіші шрифти: *"Impact", "Ravie","Microsoft PhagsPa",*

"Andalus", "Magneto", "Monotype Corsiva", "Aharoni", "Microsoft Uighur", "MV Boli", "Wide Latin", "Viner Hand ITC", "Lucida Handwriting", "Playbill", "Mistral".

Кегль шрифту обирається випадковий від 15 до 24.

Довжина тексту випадкова від 6 до 12 символів.

На холсті використовується лінійний градієнт з 5 випадковими кольорами.

Поверх літер розміщаються «шуми»: від 10 до 15 ліній.

Кольори ліній та літер обираються випадково.

Використані алгоритми динамічних перетворень «Twirl», «Spherize» та «Pyramid», метод обирається довільним чином.

Результат формування капчі представлено на рисунках 13-15.

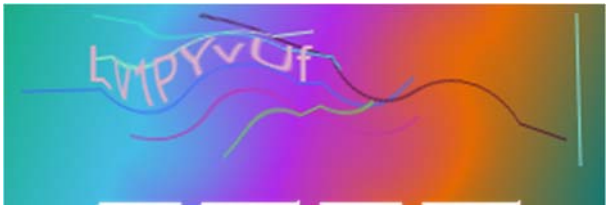

Рисунок 13 – Застосування методу «Twirl»

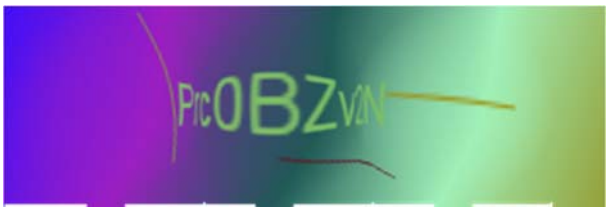

Рисунок 14 – Застосування методу «Spherize»

Рисунок 15 – Застосування методу «Pyramid»

Висновки. Досвід побудови капчі з використанням елементу canvas показав простоту та зручність реалізації. Дійти висновку, що використання капчі на веб-сторінках є досить дієвим способом боротьби з ботами-спамерами. Для запобігання автоматичного розпізнавання тексту обов'язково слід використовувати шуми, різні маніпуляції зі шрифтами та кольорами, динамічні спотворення зображень – все це є гарантією захисту сайту від спаму.

ЛІТЕРАТУРА

1. Прись І. С. Аналіз і тестування якості captcha. / Прись І. С. // Восточно-Европейский журнал передовых технологий, 2010, № 2 (46) – Т. 4 – С. 36 – 38.
2. Горло Н. Е. Подходы к построению защищенной системы управления контентом / Горло Н. Е., Пескова О. Ю. // Известия Южного федерального университета. Технические науки, – Таганрог, 2010. № 11. – Т. 112, С. 75 – 83.
3. Пиксельные искажения с билинейной фильтриацией в HTML5 canvas. [Електронний ресурс] – Режим доступу: – https://habrahabr.ru/post/138668/ – Дата звернення: 04 травня 2016.

УДК 519.816

ОПТИМИЗАЦИЯ ДИАГНОСТИКИ ТЕХНИЧЕСКОГО СОСТОЯНИЯ ГРУЗОВЫХ КАРЬЕРНЫХ АВТОСАМОСВАЛОВ

Э.Ю. Прокуда
ассистент кафедры метрологии и информационно-вычислительных технологий, Государственное высшее учебное заведение «Национальный горный университет», г. Днепропетровск, Украина, e-mail: elinka9891@mail.ru

Аннотация. В статье рассмотрена задача поиска неисправностей методом динамического программирования. Представлена диагностическая процедура поиска неисправностей с минимальными затратами.

***Ключевые слова**: оптимизация, диагностика, динамическое программирование, автосамосвал.*

OPTIMIZATION OF FINDING TECHNICAL CONDITION CARGO CAREER DUMP TRUCK

Elina Prokuda
Assistant of Metrology and Information Computation Technologies Department, State Higher Educational Institution "National Mining University", Dnepropetrovsk, Ukraine, e-mail: elinka9891@mail.ru

Abstract. The article deals with problem of searching for faults by dynamic programming. There are presented diagnostic troubleshooting procedure at minimal cost.

***Keywords**: optimization, diagnostics, dynamic programming, dump truck.*

Введение. Карьерный автосамосвал сложная система**,** состоящая из множества элементов. И при поломке автосамосвала не представляется возможным сразу определить ее причину.