mod_kaPoW: Putting the 'PoW' into Proof-of-Work!

Ed Kaiser
Portland State University
edkaiser@cs.pdx.edu

Wu-chang Feng
Portland State University
wuchang@cs.pdx.edu

ABSTRACT

Attacks from automated web clients are a significant problem on the Internet. Many web sites employ CAPTCHAs, Turing tests that hassle automated agents. Unfortunately, they require frequent human user input, can be subverted by adversaries willing to hire humans to solve challenges, and become ineffective as computer vision techniques improve.

Several alternative defenses based upon cryptographic methods have been proposed to achieve the same goals. "Proof-of-Work" (PoW) systems prioritize clients based on their willingness to solve computational challenges of client-specific difficulty. Unfortunately, few proof-of-work schemes are ever deployed since they require wide-scale adoption of special client software to operate properly.

This demonstration presents `mod_kaPoW`, a novel system that combines the efficiency and *human-transparency* of proof-of-work with the *software backwards-compatibility* of CAPTCHA schemes. `mod_kaPoW` attaches challenges to HTML links and the client's browser uses JavaScript to solve them before requesting the linked content. Accessibility is maintained for clients without JavaScript.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection – *Cryptographic Controls, Access Controls, Invasive Software*

General Terms

Security, Algorithms

Keywords

Proof-of-Work, HTML, JavaScript

1. THE SYSTEM

Our system `mod_kaPoW` [1] combines the *user transparency* of proof-of-work systems [2, 3] and the *backwards compatibility* of CAPTCHA schemes [4]. The approach uses URL rewriting, like that employed by content-distribution networks such as Akamai. The system, highlighted in Figure 1, modifies an Apache web-server to dynamically rewrite HTML links to include computational challenges. The server also sends a small piece of JavaScript code that the client's browser uses to solve the embedded challenges to create valid URLs.

The bulk of the system lies within the Apache web-server module shown in Figure 2. The module is divided into two filters; an *issuing filter* that embeds proof-of-work challenges in outbound HTML content and a *verifying filter* that prioritizes inbound request URLs based on having a valid challenge with a correct answer.

The system leverages the pervasiveness of JavaScript; when a client's browser parses a PoW-protected link, the provided script solves an equation and appends the results to the URL. Upon receiving a request, the verifying filter parses the request URL and extracts the appended PoW variables. If they satisfy the equation, the request is served by the high-priority virtual host.

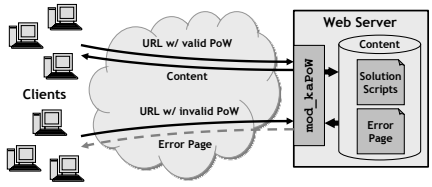

Figure 1: The `mod_kaPoW` system, with it's additions to the Apache web-server highlighted.

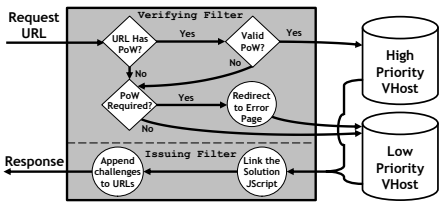

Figure 2: The `mod_kaPoW` Apache module showing the processing of a URL and the corresponding content.

Lacking a correct answer, a request is redirected to the low-priority virtual host, providing limited access to clients without JavaScript. That virtual host allows a limited number of non-persistent connections and serves lower resolution versions of the content. Some content, as specified by a configuration file, may not be available on the low priority virtual host; requests for such content result in error pages that retry the request after properly solving an up-to-date challenge.

2. REFERENCES

[1] E. Kaiser and W. Feng. mod_kaPoW: Protecting the Web with Transparent Proof-of-Work. In *Proceedings of Global Internet*, April 2008.

[2] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Proceedings of CRYPTO*, August 1992.

[3] W. Feng, E. Kaiser, W. Feng, and A. Luu. The Design and Implementation of Network Puzzles. In *Proceedings of IEEE INFOCOM*, March 2005.

[4] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Proceedings of Eurocrypt*, pages 294–311, 2003.

SIGCOMM '08, August 17–22, 2008, Seattle, Washington, USA.
ACM 978-1-60558-175-0/08/08.