Origin-Bound Captcha: an Approach to Detect and Defeat Compelled SSL Man-in-the-Middle Attack

Lei Wei

Draft of June 26, 2012. Please do not distribute.

1 The Problem

This paper concerns with the compelled SSL Man-in-the-Middle attack [2], in which government agencies compel a certificate authority (CA) to issue false SSL certificates that can be used to covertly launch man-in-the-middle attack to intercept and hijack users' secure SSL communication with SSL enabled servers. This is a well-known attack that exploits the flaw in the CA model of the public key infrastructure and has been discussed and speculated in the community for a long time. Recently there have been reports [1] that some governments used fraudulent SSL certificates issued by colluding CA to silently eavesdrop on SSL connections of citizens in their countries for surveillance and censorship purposes. Such attacks significantly undermines the security properties provided by SSL and greatly threatens the secrecy of the data communicated on the internet and privacy of its users.

2 Proposed Approach

2.1 Assumptions

Our approach requires a browser add-on to be installed on the user's machine and requires only minimal changes to the server side software. At this point we feel like it is necessary for the server side to collaborate in order to thwart such strong adversary with ISP level attack capabilities that can perform arbitrary DNS and IP hijacking attacks in the area under his control. We assume that the attacker is not able to obtain the secret key corresponding to the public key of the legitimate webserver by breaking the cryptosystem. However, he is able to obtain a certificate for a public key of his choosing different from that of the legitimate webserver's. In the following discussion we refer to a legitimate SSL enabled webserver as server and a legitimate user as user.

2.2 The Approach

We introduce *Origin-bound CAPTCHA* that aims to make large scaled automated Man-in-the-Middle attacks infeasible for an attacker, and a way for the user to detect when such an attack occurs. Traditionally, captcha is used to tell apart a human being from a machine and is commonly used to secure web authentication system against automated online attacks. This defense is not effective against a Man-in-the-Middle attack because the attacker machine simply forwards whatever it receives from the server to the user to solve, but the user is totally unaware of the fact of being attacked.

Our approach does not aim to detect the attack during the SSL handshake, but delays detection until when user needs to solve a captcha before typing in his login credential. The core insight of our approach is to make the content of the captcha bound to the identity of the server, rather than being totally random, so that the user solving the captcha is able to verify that the certificate he just accepted during the SSL handshake indeed belongs to the server he is communicating with. The captcha becomes "non-transferable" in some sense because the attacker risks being detected if he simply forwards the captcha to the user to solve, but would need to deploy a human being to solve by himself. We now describe the construction of the captcha scheme.

Suppose that the authentic certificate of the server endorses the public key PK. An origin-bound captcha (A, B) consists of two traditional captchas A and B whose contents are derived from $\mathsf{H}(\mathsf{PK}||s)$ and s, where H is a hash function and s is a short random salt. A is a captcha for s, displayed as characters in hexadecimal format. For usability reasons, we choose s to be about 20 bits (5 characters shown in hexadecimal format). B is a captcha for $\mathsf{H}(\mathsf{PK}||s)$ and only the first 6 characters in hexadecimal (24 bits) of the hash value are displayed.

We assume that a browser add-on is installed on the user's machine. The add-on records the public key PK' endorsed by the certificate that is verified during the SSL handshake process. Before the user is asked to input his login credential, the server sends the captcha to the user to solve. After the user solved the two captchas, by which we denote the two strings as a and b, the add-on obtains them and verifies if $\mathsf{H}(\mathsf{PK}'||a)_{1,\ldots,6} \overset{?}{=} b$. If it is not true, it then raises an alarm to the user. It is not hard to see that if the attacker used a PK' different from PK during the SSL handshake, with high probability the hash value would not verify. Here we have omitted some engineering issues regarding how to deal with user misidentifying the captcha which will be considered in the future.

References

[1] An update on attempted man-in-the-middle attacks. `http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html`.

[2] C. Soghoian and S. Stamm. Certified lies: detecting and defeating government interception attacks against ssl (short paper). In *15th International Conference on Financial Cryptography and Data Security*, pages 250–259, 2011.