Secure Browsing Mode

Version 1.2, 27 June 2006

Ivan Ristic (ivanr@webkreator.com)

Introduction

It is widely accepted today that web applications are inherently insecure. A lot of energy was invested in the past years into making web applications more secure, but there is only so much we can do with the fundamentally insecure foundation. This brief document proposes a set of possible browser improvements that would allow us to establish, *gradually*, a secure environment for web applications.

Goals

Main goals:

1. Reduce impact of insecure web applications by making the client devices more security-savvy.
2. Create new, well-designed, standards to replace current insecure practices.

Possible effects:

1. Eliminate Cross-Site Request Forgery.
2. Eliminate off-domain information leakage.
3. Eliminate session-based attacks (session fixation, session hijacking, session token prediction, etc).
4. Make phishing more difficult.
5. Eliminate web site spoofing (e.g. through DNS attacks).
6. Increase security in shared-browser environments.

Concept

Over the years a number of security problems have been discovered in our web application model and we have done our best to mitigate them. To design a brand new web application model that does not suffer from fundamental security issues today should not be very difficult. However, we are severely limited by the constraint of backward compatibility. The biggest challenge is to find ways to use the existing standards or evolve them and allow for a transition from the current insecure state into some future secure state.

This document proposes introduction of a *Secure Browsing Mode* (SBM). It consists of nine requirements and improvements.

F1: Backward compatibility

Secure Browsing Mode must be designed to allow the applications to take advantage of its

features when accessed by a compliant client device, continuing to work equally well with devices that do not support this mode. This requirements serves to allow the new users and those that are security conscientious to reap the benefits of SBM, but still make it possible for the existing users to continue to use SBM applications.

F2: Visual representation of the Secure Browsing Mode

SBM applications must be represented in a way that makes them appear different than normal web applications in order for the users to understand they are given an additional layer of security. *This is the same marketing trick that worked very well for SSL*. Determining the exact representation and differentiation is out of scope of this document. It's a job for usability experts to research and debate.

It is important for people to understand that only SBM gives them the security they need, especially in the initial period. For example:

1. SBM has features that prevent phishing.
2. However, it is still possible to wind up at a phishing site while in normal browsing mode. If the phishing site asks them for his credentials or credit card details the user must realise he is not secure as there are no indications of the SBM being active.

F3: Secure Browsing Mode descriptor

A SBM application may need a lot of space to describe its security requirements. It is therefore not feasible to transmit this information with every HTTP transaction. A more efficient solution would be to introduce a new HTTP header and have it point to the descriptor containing the necessary information.

At the very least the descriptor needs to contain the following information:

- Application name.
- The URI space occupied by the application (a list of domains and paths, as already used for HTTP Digest Authentication).
- Information about the organisation.

Only one descriptor should exist per application, even in the cases when the application spans multiple domains. Domains other than the domain that hosts the descriptor will demonstrate their willingness to participate in the application by emitting the header with the descriptor URI. A special resource name may be established to allow client devices to test for the presence of the descriptor header (e.g. file sbm.xml placed at the root of a web site).

F4: Control over client devices

By allowing applications to have greater control over what clients can do we can eliminate certain attack vectors. For example, a web application that does not use client-side scripting may want to disable client-side scripting altogether. By doing that it will also eliminate scripting-based attacks.

Possible features:

1. Disable client-side scripting completely.
2. Allow only scripts that are present in <HEAD>.
3. Control exactly which components/plug-ins are allowed in the application (e.g. no Flash, no Java).
4. Disable DOM manipulation.
5. Fine-grained control of the scripting capabilities: network communication options (HTTP communication), DOM inspection and manipulation, access to cookies, etc.
6. Caching of resources and cookies (e.g. allow descriptor to flush cache "on demand").

F5: Mandatory wire-level security

While SSL failed as means of establishing the identities of the parties engaging in communication, it served remarkably well as means of protecting the communication channel. Certain improvements are needed to increase robustness (SBM applications only):

1. Make SSL mandatory for SBM applications.
2. Only accept valid SSL certificates, with no exceptions.
3. Allow for certificate upgrades and expiration. (option 1: server must demonstrate possession of the previous certificate; option 2: server must authenticate using some other method (e.g. Digest)).
4. Remember certificates used in previous visits. Warn the user if the certificate changes unexpectedly.

F6: Mandatory mutual authentication

As a phishing prevention measure, all servers hosting SBM applications must support automatic mutual authentication. Compliant client devices should refuse to talk to servers that fail to authenticate. Mutual authentication should be performed using client certificates. For example, an per-application client certificate could be generated on the first visit to an SBM application.

The requirement for mutual authentication described here does not apply to any authentication web application wants to implement. This feature discusses automatic *implicit* authentication that is carried out behind the scenes for the sole purpose of identifying the site to the user.

F7: Session management improvements (optional requirement)

Session management is a required feature for all but trivial web applications. Still, this important piece of web application security is not standardised. Instead it is left to be implemented by every platform. This is the real cause of all our session management insecurities.

The following should be standardised:

1. Token generation.

2. Session timeout.
3. Idle user timeout.
4. New mechanism to transport session tokens, one that will not be accessible using client-side scripting.
5. Generate per-session throw-away digest credentials.
6. Explicit log out

Notes:

- An implementation of implicit mutual authentication (see F6) would eliminate session hijacking provided SBM applications attach application sessions to SSL client certificates..
- Allowing the application to control client devices (F4) could be used to specify idle timeout (i.e. instruct browser to flush RAM-based cookies and application history after a period of inactivity).

F8: Information Leakage Prevention

There are two requirements in this section:

1. SBM applications must not be allowed to make requests to domains/URI space outside the application. This is to prevent information leakage (e.g. as a result of XSS attack).
2. Requests from outside SBM applications must not be allowed into the application. This is to prevent Cross-Site Request Forgery.

As a side-effect, the above two would also prevent deep linking and external links. SBM applications should be allowed to make exceptions in their descriptors.

F9: You are entering a SBM application for the first time warning

We will probably need to have trust levels within a SBM application. For example, upon the fist visit there should be minimal trust. On the first visit the toolbar should say that the user has not visited the site/application before. On this level it should not be possible to transmit information to the web site. The trust level can be increased only explicitly, by the user.

We are hoping to achieve the following:

1. User arrives at a rogue web site which appears to be a web site the user is familiar with.
2. If the rogue site is not a SBM application the user should be aware there is no trust.
3. If the rogue site is a SBM application itself the user would be informed there is no established trust and should be able to determine something unusual is taking place.

Note: The aim of this feature is not to eliminate phishing, but to allow those users that are security-aware to detect a phishing attempt.

Related Work

- Content Restrictions, http://www.gerv.net/security/content-restrictions/
- User Agent Authentication Forms, http://www.w3.org/TR/NOTE-authentform

Acknowledgements

Many thanks to Amit Klein who was kind enough to review early versions of this document and point out the weak areas.