A Browser-Based Kerberos Authentication Scheme

Sebastian Gajek¹, Tibor Jager¹, Mark Manulis², and Jörg Schwenk¹

¹ Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
² UCL Crypto Group
Louvain-la-Neuve, Belgium
{sebastian.gajek,tibor.jager,joerg.schwenk}@nds.rub.de,
manulis@crypto.ucl.be

Abstract. When two players wish to share a security token (e.g., for the purpose of authentication and accounting), they call a trusted third party. This idea is the essence of Kerberos protocols, which are widely deployed in a large scale of computer networks. Browser-based Kerberos protocols are the derivates with the exception that the Kerberos client application is a commodity Web browser. Whereas the native Kerberos protocol has been repeatedly peer-reviewed without finding flaws, the history of browser-based Kerberos protocols is tarnished with negative results due to the fact that subtleties of browsers have been disregarded. We propose a browser-based Kerberos protocol based on client certificates and prove its security in the extended formal model for browser-based mutual authentication introduced at ACM ASIACCS’08.

1 Introduction

Motivation. An immediate goal browser-based protocols strive for to meet is user authentication and access control to services or private information. A widely adopted approach is to use TLS in server authenticated mode and execute a protocol on top, where the user enters a password in a Web form. To this end, the user has to memorize a plethora of passwords. The problem with passwords is that the user frequently forgets about them. Otherwise, it would be unnecessary to include a "Forgot your password" link in a Web application). Furthermore, the user tends to recurrently choose the same low-entropy password, thus making offline dictionary attacks feasible. In order to alleviate the problem, 3-party authentication protocols have been introduced where a trusted third party is asked to issue a token that is valid for a fixed time period and permits access to some service. A pioneer and quite successful protocol for closed networks that emulates the task is the widely adapted Kerberos protocol [1]. Here, the Kerberos server issues an token in form of a ticket that the client may redeem to authenticate to a server. Related protocols that adapt the idea are Microsoft’s Passport and its successor Cardspace, the Security Assertion Markup Language (SAML), the Liberty Alliance project, the Shibboleth project...
for university identity federation, or WS-Federation, whereby SAML is an open protocol standard and basis for Liberty and Shibboleth.

The migration of the Kerberos idea to open networks, in particular, the Internet is peppered with problems (see Section 2). In particular, the problem with the browser-based realization is that some assumptions have been made which are unfounded today. Most notably, the user is assumed to determine the authenticity of a Web server on the basis of server certificate and the Domain Name System (DNS) is assumed to be an authentic host name resolution protocol. The first clashes with usability studies, showing that the average Internet user neither understands server certificates nor perceives security indicators in commodity browsers [8,23]. The latter is a crucial factor for the enforcement of the **Same Origin Policy (SOP)**. This security policy, which is universally supported by browsers, loosely states that Web objects are accessible by other Web objects under the condition that they are from the same domain. However, many attacks against the domain name resolution exist, ranging from Javascript code that alters a router’s configuration [22] to large scale DNS attacks [15]. A related attack vector arises from cross Site scripting (XSS) attacks [16] where the adversary injects some malicious code into the response of the application server. Since the code is in the same security context, the SOP does not apply. Consequently, malicious code can break free and invoke arbitrary browser scripting functionalities.

**Our Contribution.** We solve the above problems by presenting a Browser-based Kerberos-like protocol, in the following denoted by BBKerberos, that is close to the native Kerberos scheme. Our BBKerberos protocol:

- combines authentication with key agreement: The user authenticates to the Kerberos server through a TLS client certificate in addition to (optional) passwords. The Kerberos server issues an authentication ticket for the application server which is concealed within an HTTP cookie. The cookie is transferred in another TLS session whereby the browser authenticates to the server using the same client certificate. Thus in both TLS connections key agreement is linked to authentication through the client certificate.

- binds the Kerberos ticket to a specific browser. The ticket is linked to the client certificate. Thus, attacks that enable adversaries to extract the cookie carrying the Kerberos ticket (e.g. XSS, Pharming) work. However, the attacker is now unable to use the cookie. The reason is that the application server learns from the underlying TLS protocol session that the client is a legitimate owner of the client certificate (note that in the TLS protocol the client authenticates to the Kerberos server by signing a protocol transcript and proving ownership of the corresponding private key). Here, we make use of the fact that any feasible adversary does not have access to the long-term secrets for the TLS layer. It has only access to secrets on application layer. Conversely, the application server may extract the public key from the TLS layer and verify the cookie.

- provides secure single sign-on. The sign-on ticket may also be reused in a federation of application servers. Application servers need to establish a