Countermeasures for Protecting Legally Sensitive Web-Powered Databases and Web Portals

Theodoros Evdoridis
University of the Aegean, Greece

Theodoros Tzouramanis
University of the Aegean, Greece

INTRODUCTION

The issue of the escalation of security breaches in the field of Web systems has caused a great deal of disquiet in the computer security community. The majority of recorded security violations against legally sensitive portals have raised numerous issues both at an individual and at an organizational level. Furthermore, taking for granted the fact that security achieved through the isolation of the targeted systems is a path which no one is willing to follow, it is understood that security countermeasures must be perceived and applied without any alterations in respect of the current operational scheme. The economic and social reasons for using the Internet are still far too compelling (Schneier, 2005). Looking in this direction, the complexity as well as the urgency of the present situation has attracted specialists from other scientific sectors, such as psychology and law, who contribute to the search for an integrated multilevel solution required in this context.

BACKGROUND

The issue of making computers that host legally sensitive information secure has been a major concern of the computer security community over the years (Computerworld.com, 2003). A group of experts argue that security features should not be built into the Web portal's or into the Web database's infrastructure, but rather added on to it, according to emerging needs, because doing so would increase dramatically the system's complexity, rendering it cumbersome to debug, to maintain, and to further develop. Another view is held that claims a mixed solution must be adopted. As routine tasks like access control must be handled in the database and because new threats emerge daily, add-on security solutions should be applied when it is considered necessary.

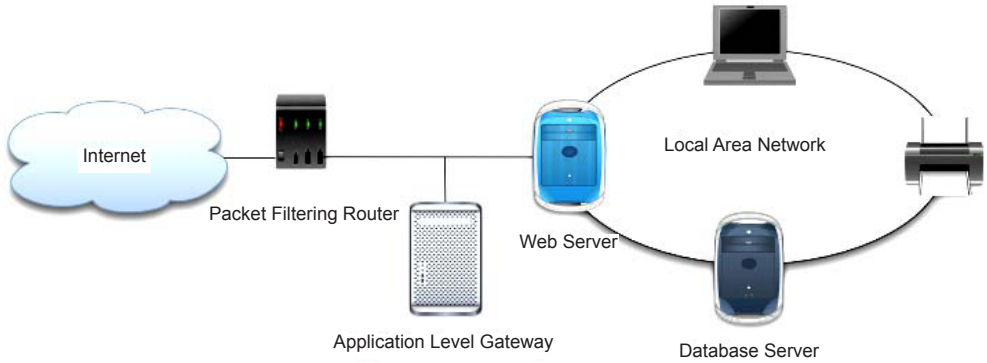
MAIN THRUST

It is obvious that in view of blocking any possible attack (see other sections, for example, on the "Security Threats in Web-Powered Databases and Web Portals," which also appears in this publication), a corresponding multilevel countermeasure policy must be followed. Below, the most common security countermeasures for these types of attacks are reviewed.

Network-Level Countermeasures

Looking forward to preventing all possible attacks performed on the network layer of a legally sensitive Web portal, security mechanisms must be implemented (Microsoft Corporation, 2003). Primary solutions for these type of attacks are cryptographic protocols, such as SSL or TLS, that undertake the task of encrypting communication data from the client to the server, and vice versa. The usage of these protocols guarantees that the data are revealed only to authorized parties, thus ensuring information confidentiality. Furthermore, by adopting Ipsec, which is an obligatory part of Ipv6 (Wikipedia, 2006), additional security mechanisms that ensure authentication, data confidentiality, and message integrity between communicating parties are interpolated in the security scheme. As a result, sniffing attacks, while successful in capturing the data, fail in reaching their goal, as the captured data are in a encrypted form that cannot be used alone to produce their decrypted version. As for tampering, message authentication codes included in Ipsec can be used to discover if the received message is really the original one sent the legitimate sender (Tipton & Krause, 2004). In addition, the message authentication code included in the above cryptographic protocols by using parameters that are related with current time, assures that no prior connection can be used to forge a new one, thus preventing any session

Figure 1. A firewall protected Local Area Network containing the Web portal assets



high-jacking attempt. Finally, to successfully counter the spoofing threat, access control mechanisms are needed such as firewalls, both network and application ones, that have appropriately been configured. The first category, known as packet filtering routers, is responsible for reading packet headers and deciding, according to a given access control list that expresses the security policy that needs to be enforced, if the packet should be forwarded or blocked. One of the most commonly encountered application level firewalls is application-level gateways. These systems serve as proxy servers and act as a relay for application level traffic.

Host-Level Countermeasures

For defence against these kind of threats, all core components of the Web portal must be running the latest stable versions, including service packs, security updates, and patches that fix bugs or render the program, of all software components that they utilise, more reliable. Additionally, specialised virus detection software should be active at all times, looking for the presence of already known viruses on the hosts and the network. Apart from this, all default accounts on operating systems and servers should be deactivated and all external connections to the intranet where the Web portal's hosts lies must be forced to pass through network and application level firewalls (Oppliger, 2002). Proxy server usage is also recommended because this network service forces all connections and requests to be made against a third computer system that, in turn, performs requests on behalf of the client to the default server. This schema adds another security layer because information, such as ip addresses of the hosts, are hidden from the client, and direct access to the hosts is prevented. In conclusion, intrusion detection systems must be adopted in order to detect all types of malicious network traffic and computer usage that can't be detected by a conventional firewall (Wikipedia.org, 2006).

Application-Level Countermeasures

To be able to counter attacks performed against the application software, developers must pay heed when designing the interfaces that are going to be used for user-submitted data (Microsoft Corporation, 2003; Splain, 2002). To prevent the buffer overflow threat from coming to pass, data validation regarding its size must take place. To be able to prevent SQL injection attacks at an application level, the routines for constructing dynamic SQL statements must be modified to exclude special characters such as ";" and the application should connect to the database with least-required privileges (Breidenbach, 2002; Su & Wassermann, 2006).

Besides, when HTML forms are disposed for authentication, data including usernames, password, and cookie should be transmitted via SSL in encrypted form as a single entity. This technique prevents credential disclosure and cookie replay attacks, because the attacker would not be able to sniff the cookie out, as all traffic is encrypted. Likewise, client software and operating systems need to be up to date to avoid any vital information leakage on this side. Moreover, software developers should embed password evaluation routines on the application, forcing users to use passwords that comply with minimum-security standards.

Finally, in order to protect against cross-site scripting attacks, site owners must never trust user input and always filter metacharacters. This will eliminate the majority of XSS attacks. Converting "<," ">" and other possibly malicious characters to their HTML equivalents is also suggested when it comes to script output (Cgisecurity.com, 2003). Figure 2 depicts a malicious link specially crafted to take advantage of the problematic site "subject_to_xss_site.com" and gain access to the victim's cookie.

When employing character conversion on specific characters, the once malicious link is no longer a threat because it can't be correctly parsed to produce the expected results for the attacker.