PLANNING ASSUMPTION

September 5, 2000

Securing the Extranet Web Application

Steve Hunt

Giga Position

Business managers across all industries are rolling out Web-based applications to their remote users, distributors, suppliers and business partners. Unfortunately, they often make the applications available before they take reasonable security measures. Time-to-market pressures and a lack of familiarity with the inherent risks of Internet applications are generally to blame. As a result, companies are bearing more risk with extranet applications than they would with internal applications. Properly securing this kind of extranet architecture involves authenticating users, establishing a secure session, protecting the content on the server, defending against server hacking, enhancing the connection to back-office systems and adding access-control measures to specific data repositories.

This Planning Assumption will address network security architecture, leaving application development security issues for a separate document.

Proof/Notes

A Web-based application that permits connections from untrusted, external networks (like the Internet) and also reads or writes to a trusted data repository inside the local area network (LAN) poses challenging security architecture problems. The good news is that everyone is in the same boat these days.

The security architecture for this basic extranet design includes six dimensions:

1. Authenticating the user
2. Securing the session between the remote user and the Web server
3. Protecting the content of the HTML pages
4. Hardening the Web server operating system
5. Securing the link between the Web server and the back-office data repository
6. Properly configuring access-control rules on the data repository

Let's look at these dimensions one at a time.

1. Authenticating the User

Normally thought of as simply logging in, authentication can take several forms. The most common, of course, is that the user is prompted by the application to present a user identification and password. When the application requires a greater degree of confidence in the identity of the end user — during high-value transactions, for example — two-factor authentication using tokens is standard. Passwords are easy to use and deploy, but they are not reliable. Users write them on notes, share them with friends and make them easy to guess. Tokens mitigate those shortcomings by requiring that the user physically carry the device, and, in place of a password, enter a one-time numeric code. **RSA Security** has the largest market share of tokens with its SecureID card system.

Smart cards and digital certificates will replace passwords and tokens as the dominant authentication type for online applications in 2003 [.9p]. Passwords will be a subordinate authentication method for localized, low-value applications. Tokens will remain in use for specialized applications through 2007 [.8p]. Today, token providers, such as **Activcard**, are already building the capability to support passwords, tokens and smart cards simultaneously. With such an infrastructure, a company can be an early adopter of newer authentication types within some user populations or business units, while still supporting a status quo among other users.

Any authentication type requires an infrastructure to support it. The infrastructure is defined by processes to register, suspend, revoke and share credentials. Companies are advised to implement an authentication infrastructure that supports as many authentication types as practical.

Beyond supporting the authentication type, an authentication infrastructure may also manage user privileges to applications. **Netegrity** and **Securant** are market leaders in this space (see Planning Assumption, Single Sign-On 2000: The State of the Market, Steve Hunt, for more details**).**

2. Securing the Session

Hosting a Secure Sockets Layer (SSL) session from the Web server to the end user is a common and fairly straightforward use of a certificate (from **VeriSign** usually) and SSL software (from RSA Security usually). There are many resources available to assist your technical staff in setting this up. By virtue of hosting SSL sessions, you have a virtual private network (VPN) that will keep the traffic private as it travels across the Internet. SSL comes in two strengths: 48-bit and 128-bit — 48-bit is suitable for securing time-sensitive information of moderate value; 128-bit is suitable for high-value transactions up to, say, $50,000 in redeemable value. For higher value transactions, a mutually authenticating VPN is required. (See research by Giga analyst Jim Slaby for more on VPNs.)

3. Protecting Content

Remarkably, even after several high-profile hacks of Web sites, where the content was defaced (I'm thinking of the US National Security Agency's and *The New York Times*' sites, in particular), many companies still forget to secure the Web server content. There are methods a hacker may employ to change not only how a company's Web content appears, but also more subtly to poison or pervert the content. Imagine if a company is displaying inventory levels, sales figures or catalog prices. A hacker may make minor changes to the figures — not immediately noticeable to the administrators or end users — with devastating results. Investors may make trades based on the fraudulent information, or supply chain partners may increase or decrease production, etc. The obvious remedy is to somehow prevent unauthorized content modification. Unfortunately, such protection cannot be completely effected by common security measures like a firewall or better application development. Many of the threats to content integrity are inherent in HTML itself. There are only three effective courses of action:

1. Run all content from CD-ROMs. A CD-ROM cannot be modified, so the integrity of the data is assured.
2. Lock down the server so that while in production it is essentially a read-only device. **Qiave** makes a simple-to-use software upgrade to an NT or a Solaris Web server that prevents writing to the disk while in locked mode (www.qiave.com).
3. Install a proxy from **Sanctum Inc**. (www.sanctuminc.com). This box will not only inhibit writing to the disk of the Web server, but it will also protect against nagging threats like URL modification, PERL scripts in the search field and other common annoyances in Web security.

4. Hardening the Operating System

If the Web server is permitting promiscuous connections to the Internet, even if there is a firewall, the Web server will be hacked. It is a certainty. To reduce the damage a hacker may do, the Web server operating system must be hardened. That means the kernel, or brains of the system, will be secured in such a way that a hacker cannot perpetrate damaging hacks. Operating system hardening is detailed in many technical books and online sources. Most system administrators will know how to find them. The process is basically stripping down the functionality of the server so it is dedicated only to mission-critical tasks. For example, the FTP service will be disabled, and miscellaneous system IDs and passwords will be eliminated.

5. Securing the Link

Since hackers, along with the entire world, have access to the Web server, it is important that they not have access to the rest of the network, especially the critical back-office data repository. Securing the connection from the Web server to the back office is crucial. The firewall plays an important role here. It will be configured to filter traffic between the two hosts by source IP address (the Web server), destination IP address (the database), port and service (e.g., SQL). That means that even if hackers have taken control of the Web server, they still cannot roam the network. The catch is that they have complete access to the database as long as they use only the SQL service through the determined port. That's a lot of exposure right there.

The solution is to add proxies. Any proxy technology is better than none, in that the proxy performs transactions on behalf of another server — in this case, the Web server. **Kyberpass** is one specialized security middleware device that serves the purpose of a proxy (www.kyberpass.com). It watches all traffic between the two devices, looks for anomalies in the SQL calls and uses granularity to restrict access to the database. A "proxy on steroids" solution is available from **Whale Communications**. (For a description of the company's e-Gap, see IdeaByte, Solution to the Swiss-Cheese Firewall, Steve Hunt.) Basically, the Whale solution filters traffic according to rigorous and detailed access-control lists while also logically separating the back-office resources from the Web server. That is, instead of a TCP/IP connection between the hosts, the e-Gap uses a high-speed SCSI connection between the exterior and interior hosts. That way, there is no possibility of executing any surreptitious or malicious connection. Whale's e-Gap is the ultimate proxy between two hosts.

6. Configuring Access-Control Rules

Oracle 8i is the most advanced database when it comes to enforcing granular user or role-based access controls. However, any database will have some access-control capabilities. DB2 benefits from the mainframe security system RACF. Access control on Unix and Windows NT systems can be enhanced with eTrust Access Control from **Computer Associates**. But if your back-office data repository is an enterprise resource planning (ERP) application, the access-control options are more limited. In any case, work with the application support team to ensure that connections from the demilitarized zone (DMZ) cannot exceed appropriate privileges.

Alternative View

The public key infrastructures (PKIs) sold by **Entrust**, RSA Security and others tout capabilities that include the seamless connection of external users to internal resources. Therefore, the degree to which those capabilities may actually be delivered may supplant and replace many of the security measures mentioned here. It is possible that a transaction-based security infrastructure built on a PKI could preclude the need for many system security measures. In this model, all security measures (including firewalls, switches and access-control software) would identify the user and grant access privileges by recognizing the certificate accompanying each packet.

Findings & Recommendations

By not taking appropriate security measures, companies are bearing more risk with extranet applications than they would with internal applications. Properly securing a Web server or any external application is a matter of taking the following reasonable security measures:

1. Authenticating the user in a way that matches confidence in identity with the value of the transaction
    a. Use passwords for low-value transactions; use tokens or smart cards for higher-value transactions; seek a single authentication infrastructure for managing multiple-authentication types.
2. Securing the session between the remote user and the Web server using SSL or VPN technologies
    b. SSL is suitable for modest-value transactions (up to $50,000 in perceived value); VPN and authorization controls are necessary for higher-value transactions.
3. Protecting the content of the application server using write-access control or content proxies
    c. A content proxy such as AppShield from Sanctum is recommended for broad and flexible content protection.
4. Hardening the Web server operating system to protect the integrity of the system kernel and file system
    d. Use standard hardening methods to remove unnecessary system IDs, update operating system patches, remove unneeded services, tighten file system access control and improve passwords.
5. Securing the link between the Web server and the back-office data repository with proxies or an "air gap"
    e. Utilize Kyberpass for SQL transactions and the e-Gap from Whale Communications for all other connections between the DMZ and the back office.
6. Configuring access-control rules on the data repository properly by using native access-control features or resource access-control software
    f. eTrust Access Control from Computer Associates can enhance the native authorization facilities of Unix and NT servers.

Most of these measures are accomplished fairly quickly, and many resources are available to assist an IT staff in implementing them. So there is no excuse for bypassing them. By shortcutting any one of these, unauthorized access to a Web-based application is a much higher risk.

References

Related Giga Research

(Planning Assumption, Single Sign-On 2000: The State of the Market, Steve Hunt**)**

Planning Assumption, Recommendations for Secure E-Business, Steve Hunt

Planning Assumption, Authentication Is the First Step Toward Secure E-Business, Steve Hunt

Planning Assumption, Web Security Suites Fall Short, Steve Hunt

Planning Assumption, Extranets Drive the Security Trends for 2000, Steve Hunt

IdeaByte, Kyberpass and Genuity Are More Different Than Alike, Steve Hunt

IdeaByte, Database Encryption Is Part of an Extranet Security Architecture, Steve Hunt

IdeaByte, Securing Web Content, Not Just Web Servers, Steve Hunt

IdeaByte, Is SiteMinder Appropriate for Distributed Web Servers? Steve Hunt

IdeaByte, Single Sign-On Has Many Faces, Steve Hunt

IdeaByte, Selecting Security Products That Are HIPAA-Compliant, Steve Hunt

IdeaByte, Reliable Security Between Applications and Databases, Steve Hunt

IdeaByte, Entrust Buys enCommerce, Snubs Securant, Steve Hunt

IdeaByte, Web Single Sign-On Provides More Functionality, Steve Hunt

IdeaByte, Passwords Are E-Commerce Friendly, Steve Hunt

IdeaByte, The Four A's of Secure E-Business, Steve Hunt

IdeaByte, Solution to the Swiss-Cheese Firewall, Steve Hunt

IdeaByte, When It Comes to Stopping Web Site Hackers, Perfecto Is Perfect, Steve Hunt