GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Things To Consider
When Setting Up Form Based Authentication

Matt Powers
GIAC Security Essentials Certification (GSEC)
Practical v1.4b Option #1
August 2003

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Abstract

The Internet is still growing. An increasing number of businesses are choosing to implement web based applications to provide more convenience and increase productivity, allowing employees to access work assets from any computer connected to the Internet at any time of the day. Companies are even developing web applications that run within their own intranet. This level of access to valuable information poses significant risks and should not be taken lightly.

Authentication is the first line of defense in most web applications. It is important to ensure adequate protection is provided to a company's assets or a customer's assets. This paper discusses topics that should be addressed by developers and management when designing form-based authentication for a web application. This paper will establish a baseline for developing strong form-based authentication.

Creating web-based authentication for a web application can be a daunting task and it should not be undertaken without considering the issues involved. There are many approaches to securing a web application that can be used to avoid the most common attacks. The approaches presented within this paper should be considered based upon cost to implement and acceptable risk factors when determining if they are an appropriate solution to your authentication needs.

Authentication is the first line of defense

According to searchSecurity.com definitions, "Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be" [6]. With regards to web applications, web based authentication is validating who a user or entity is to your application across a network.

There are several commercially available products on the market that are designed to perform web-based authentication. There are risks associated with trusting third party software to manage authentication on your system, but it may be more cost effective or more time effective to employ a third party application designed by someone with more experience in centralized authentication implementation. For those companies who either do not trust third party solutions for this task, or for other reasons choose to implement it themselves, several elements should be considered.

Encryption

One of the greatest weaknesses in protecting data is assuming that no one will ever notice our data amongst the vast amounts of data traversing the Internet

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

every second. It is fairly easy for someone to find a company's information if it is not encrypted, especially if they are targeting their attack. It is best to assume that information will be seen by someone that is not supposed to have access to it if it is not encrypted.

Encryption is required to provide data protection. Transmitting company or customer information across the Internet unencrypted makes the company a very easy target. SSL has become the most common way to encrypt information on the Internet. Certifying Authorities provide server certificates that can validate the company to the users of their web applications. Users are also able to get client certificates that can be used to validate a user to a system.

Two types of Authentication

Authentication takes place in two ways within a web application. The initial login of a User to a system provides authentication of a user to the system. Each additional page requested by a user after they have logged in is usually done by authenticating some type of session key, or session id to the system. The session id is used to link the session to the user's information for each subsequent page request.

Authenticating the User

There are several types of well-known exploits that can be used against a web login form (or any other part of a web application). By the nature of the way a web browser works, it is nearly impossible to guarantee that information sent to a web application from a client will not be harmful, even when coming from a form it generated.

Even large corporations developing web applications with the intent of subverting attacks can overlook simple checks that should always be performed when developing web applications. On Oct. 22, 2002, eWeek launched its OpenHack4 contest. Oracle and Microsoft both created web applications that were setup for this contest to see how hardened their web applications were. Within 2 hours and 20 minutes Jeremy Poteet had exploited two cross-site scripting exploits found in Oracle's web application. His first exploit was possible due to a lack of parsing the user ID field from the form. More information on this specific event can be found at http://www.eweek.com[7].

Within "The Ten Most Critical Web Application Security Vulnerabilities" published by The Open Web Application Security Project (OWASP), five of the vulnerabilities listed can be attributed in part or in whole to improper handling of invalid data submitted by forms. These five exploits include Invalidated Parameters, Cross-Site Scripting (XSS) Flaws, Buffer Overflows, Command Injection Flaws, and Error Handling Problems[2].

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Form Validation

In the article "Developers play vital role in web app security" by Mandy Andress, Mandy states very clearly "developers should learn never to trust incoming data."[5] Validating form input values on the server side is one of the most important things that should be done to secure any web application. The validation of input values should be done thoroughly and should be used to sanitize all input field values within the web application.

Validation of input can be broken down into three common approaches:

1) Acceptance of only known good data - defining what good input is such as saying it is the set of all alphanumeric characters.

2) Rejecting known bad data - this is difficult to do since there are so many ways for a hacker to format bad or harmful data. This type of checking should include searching for embedded SQL statements and javascript statements that could end up being executed on the server.

While it is a good idea to search for "<SCRIPT>" tags or "javascript:" within a tag attribute, there are many ways for a hacker to format tags and attributes in a manner that they would not be matched. For example, if

```
<img src="javascript:void(0);">
```

were passed in, it would be caught, but if

```
<img src="jav
a
scr
ipt:void(0);">
```

was passed in, it would not necessarily be caught by a check for "javascript:", but it would executed in the same way by a browser. The use of Unicode characters to hide javascript commands can also be used to disguise an attack.

3) Sanitization of bad data - this is very important if your error message screens display the bad or harmful data that was entered. Sanitization of data can be difficult as seen above when attempting to reject known bad data. It is very important to sanitize data that has been rejected, especially if that data is intended to be sent back to the user as part of any page being displayed. If embedded code is caught and then displayed without being sanitized, the hacker has defeated your rejection of the known bad data.

Cross-Site Scripting Flaws

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Cross-site scripting is a fairly simple attack to perform on a web application. It entails submitting values within form fields that contain HTML tags that will execute javascript if they are not filtered out before being used by the web application. Thwarting this attack can be done through sanitization of all input. Most cross-site scripting attacks can be avoided by replacing the following characters “<”, “>”, “(“, “)”, “#”, and “&”, with corresponding HTML escape sequences “<”, “>”, “(”, “)”, “#”, and “&” respectively.

Buffer Overflow

Buffer overflow attacks are less likely to be used against custom web applications due to the underlying code not being as readily available to the attacker, unless they are attacking from inside the company. They can still sometimes be used as a denial of service attack to cause the web server to crash, but it is difficult to exploit these to execute arbitrary code without the underlying source being available. Preventing buffer overflows within a web application is performed by thoroughly testing the length of the input being provided by all form elements -- even combo box input can be spoofed by a hacker.

Command Injection

Command injection, like Cross-site scripting, is the inserting of malicious code within an input field of a form. If this code is blindly run by the web server, then the attack will be successful.

The most common use of this attack is SQL Injection. This attack involves the malicious formatting of a value that a web application passes into an SQL statement. These are often commands that will cleverly end the intended SQL statement and then execute requests for information, updates and deletes. This type of attack can be very costly to a company when they are used to compromise an important database. This is also a likely attack against a login screen, since account information is often stored within a database; it is likely that the username and password fields are sent directly to the SQL statements that are attempting to authenticate the user.

To prevent this type of attack, filtering input for SQL commands or statement constructs can be used to ensure those values are not a part of the content being submitted. Although this can be diverted, SQL values can also be surrounded by two single quotes so that their content is interpreted completely (with the exception of two single quotes…). The use of stored procedures and prepared statements can also help prevent this type of attack.

Error Handling

When developing a web application, it is important to handle all exceptions and errors securely. Hackers can often find out vital system information by creating

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

intentional errors within the web application when detailed messages are presented.

Error handling should be complete and robust for developers, but the messages returned to users should be short and simple – do not provide any important information and do not display user provided information. The errors should also be logged for tracking and troubleshooting.

Client Side Cache

When authenticating a user, it is important to not use the form GET method for passing a user's name and password. The GET method will not only send the username and password as part of the URL, but it will also cache the username and password on the user's disk, in the web server logs, and any proxy server in between. The POST method should always be used when transferring this type of information to avoid having usernames and passwords cached to disk. This is a very simple thing to implement within an application and should be consistently utilized when developing web applications.

Authenticating the Session Id

While secure session implementation is beyond the scope of this paper, a session id or key must be given to the client in some fashion so that the session can be tracked by the system the next time the user accesses a page. This avoids the need for the user to have to login every time a new page is requested from the server. For security purposes it is important to not store any more information about the session itself than this key on the client side while storing the valuable data on the server.

- The transport media for communicating this id must be secure (i.e. SSL).
- Cookies are commonly used for storing the session key or session id. There are several things that should be done to make sure the session data is not easily obtainable by an attacker.
- Store only the session id within the cookie. This keeps all user information and state management on the server.
- Use long randomly generated strings for the session id. This will make it difficult for a hacker gain access to the session. If a hacker gains access to the user's session data, they can take over the session with all the roles and privileges granted to that user.
- Set the expiration values for session low. This reduces the amount of time available to an attacker if they are successful in acquiring a session.
- Change the session id often especially during major transitions such as when the user's information is updated, especially when a password has been changed.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The Employee

The most overlooked aspect of security within a company is the employee. According to Symantec's "Internet Security Threat Report VIII" more than half of the incidents that Symantec responded to for the 3rd and 4th quarter of 2002 involved computer misuse and abuse by employees. The amount of damages reported for these incidents was "significantly greater than that caused by external breaches". [4]

Symantec goes on to report that they personally witnessed "cases of highly organized corporate espionage… and even one case of email misuse that prompted criminal charges." The incidents usually did not involve the employee hacking into the system internally. The employee in most cases already had the privileges required granted to them, notably system administrators "were often the guilty party". [4]

These findings demonstrate the importance of managing employee accounts properly. Employees should only be able to see and perform actions within a system that are necessary for them to complete their job. Keeping up with the maintenance of accounts must be done diligently.

Since employees must be trusted to some degree to be able to perform their job, it is necessary for companies to put more effort into educating their employees about security issues. With regards to authentication, the employee is the responsible party for deciding what their password will be and ensuring that the password is not misused or given away to anyone else. Without a good understanding of the responsibility that this presents to the employee, they may not be diligent in keeping their password secure.

Password Protection

According to a document entitle "Strong Authentication" by WAVE, "Credentials that authenticate an individual (or entity) can take many forms and can involve something the individual … has, … knows, … or is"[8]. With regard to form base web authentication the most readily available and used form of credentials would be the password. The password falls into the category of something an individual knows. This means that it is also something that an individual can tell someone, write down, or even forget.

The User's password is the key to the system. Users don't like to use hard to remember passwords; they use what they can recall easily. Logging into a system is often considered a hassle in the eyes of the employee who just wants to get their job done. There is a flaw in the trust relationship between a system and a user. The system has to trust the user when the user to protect key to the system.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The responsibility of keeping their password a secret and the creating a strong password are the two main issues that should be need to be addressed during system design. These issues greatly affect the security of the system and are both integral to keeping authentication secure.

Enforcing strong passwords

To enforce strong passwords, companies must decide on a password policy. Password policies must then be enforced at the application level to guarantee compliance. Password policies should include minimum character length for passwords, specific requirements for passwords to contain mixed case, numbers, and special characters and the minimum number of each element to ensure the password is strong.

Minimum character length, mixed case alphanumeric, and special character requirements go a long way in prolonging brute force attacks. The amount of time it takes to break a password grows exponentially as the size of the password and the number of possible characters is increased.

Guarantee hard to break passwords

There are many tools available for cracking passwords. Attacks from inside the company can easily lead to compromise of username and encrypted password sets that can then be brute force attacked by applications such as crack. Attackers will often use a more refined attack involving a dictionary word list to attempt to find the weaker passwords on a system. The best technique for making this more difficult is using a similar check when passwords are created and requiring the user to use a different password when a match is found that would easily be recovered using crack. Although there would be too many dictionaries to implement within the system to cover all possible word-list based attacks, using the basic dictionaries and any others that may seem relevant to the company or employee environment is a great start. This will assist in guaranteeing the users create stronger passwords. The passwords would still be susceptible to brute force attack, but by increasing the minimum number of characters and the requirements on using special characters, the amount of time it takes to break the passwords using brute force can be increased exponentially.

Implementing Strong Passwords

According to “Global Information Security Survey 2003” by Ernst & Young LLP “Only 29% of organizations list employee awareness and training as a top area of information security spending, compared with 83% of organizations that list technology as their top information security

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

spending area."[3] This survey along with the information from Symantec's survey mentioned earlier seem to point to a problem with companies not putting enough emphasis on educating their employees.

Knowledge and education play a key role in strengthening the user against the social engineering attacks -- those attempting to retrieve password information from employees. Educating users of the need for strong passwords and the threat of corporate espionage can be very beneficial.

Educate users on how to create strong passwords

The responsibility of the user to create their own secure password can be simplified by teaching them techniques that help in remembering strong passwords. For example, passwords should be phrases, not just one word, that are easy to remember. Letters or spaces should be replaced with special characters, numbers, or words can be misspelled intentionally. This creates a much stronger pass phrase that is easier to remember than a randomly generated string. Another common approach is to create a password based upon a pattern it creates on the keyboard when typing it in, not based upon the keys being used. This can be used to create a mental image of the password that is easier to remember than the characters involved.

Keeping strong passwords safe

Companies often have many separate applications each requiring a different login to the system. When employees are required to create, remember, and change passwords for many systems, they often end up using the same password across many systems, which if compromised, can grant unauthorized access to many systems within a company instead of just one. Having multiple passwords also promotes the writing down of passwords so that users don't forget their password. This is a very obvious flaw in security, and can lead to vulnerabilities from unlikely sources like cleaning crews finding login information under a keyboard and being used in corporate espionage.

One way to combat this issue is by using a product that allows users to store multiple passwords in a secure manner. Password Safe is one such free tool for managing multiple passwords that can be downloaded at http://passwordsafe.sourceforge.net[9]. The only password that a user would have to remember is the password to open their profile within Password Safe. Password Safe stores the passwords encrypted for retrieval by the user if and when they do not remember a password. This is much safer than a user storing their passwords somewhere in plain text.

If an attacker gains access to the user's encrypted passwords and has some mechanism to decrypt them, the attacker would have access to all the user's

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

passwords. While this is a risk, it is not as probable as the risk of a password being stolen if it is stored anywhere in plain text.

Conclusion

There is a lot of planning that is required when developing web authentication. The key to preventing attacks is staying informed, secure programming, and user training. Information is available but some of it is either incomplete or incorrect, which is why it is best to use specific trusted sources for staying up to date on current security issues. One of the best sites available is the Open Web Application Security Project or OWASP at http://www.owasp.org/.[1]

© SANS Institute 2003, Author retains full rights.

References

1. “Guide To Building Secure Web Apps 1.1" Sept. 11, 2002.
URL: http://www.owasp.org/documentation/guide/1.1/index
(Aug. 27, 2003)

2) “OWASP Top Ten Web Application Vulnerabilities” Version 1.0. Jan. 13, 2003
URL: http://www.owasp.org/documentation/topten
(Aug. 27, 2003)

3) “Global Information Security Survey 2003”
URL:
http://int.sitestat.com/ernst-and-young/international/s?GISS&ns_type=pdf
(Aug. 27, 2003)

4) “Symantec Internet Security Threat Report Attack Trends for Q3 and Q4 2002” Report 3, Volume 3. Feb. 2003
URL: http://www.securitystats.com/reports/Symantec-Internet_Security_Threat_Report_vIII.20030201.pdf
(Aug. 27, 2003)

5) Andress, Mandy. “Developers play vital role in web app security” April 5, 2001
URL: http://archive.infoworld.com/articles/tc/xml/01/04/09/010409tcwebsecsb.xml
(Aug. 26, 2003)

6) “SearchSecurity.com Definitions”
URL: http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci211621,00.html
(Aug. 27, 2003)

7) Dyck, Timothy. “Crack in OpenHack” Oct. 25, 2002
URL: http://www.eweek.com/article2/0,3959,648540,00.asp
(Aug. 27, 2003)

8) “Strong Authentication”
URL: http://www.wave.com/about/datasheets/strongauthentication.pdf
(Aug. 27, 2003)

9) Password Safe
URL: http://passwordsafe.sourceforge.net
(Aug. 27, 2003)

© SANS Institute 2003, Author retains full rights.