Alex Colson

Dr. Lunsford

Information Security Management

10 July 2007

Controlling Website Account Information

A recent survey done by Privacy Rights Clearinghouse shows that in the past five years 27.3 million Americans were victims of identity theft. The total cost of fraudulent activity in the year 2006 was $55.7 billion. A recent study by Symantec shows that the value of a stolen credit card can be bought for as little as $1 on the underground markets. A complete identity, including date of birth, US bank account, credit card, and a government issued identification number, can be purchased from $14 to $16 (Krebs 2007). Even when websites don't collect personal information, they can still hold useful data that helps a hacker get to their goal. Web 2.0 sites, a term that is used to describe social networking over the Internet, are being developed constantly and typically require usernames and passwords to keep track of users (Lassila & Hendler, 2007). These sites are often the target of attacks making security a top priority on these types of sites (Henry, 2007). When creating a website that requires authentication, the designer must keep in mind that passwords should be stored in an encrypted format. There must also be a password policy set before launching the site; this could include the password requirements as well as how the website and webmaster

Item	Price
United States based credit card with card verification value	$1 - $6
United Kingdom based credit card with card verification value	$2 - $12
An identity; including US bank account, credit card, date of birth, and government issued identification number	$14 - $18
List of 29,000 emails	$5
Online banking account with a $9,900 balance	$300
Yahoo Mail cookie exploit - advertised to facilitate full access when successful	$3
Valid Yahoo and Hotmail email cookies	$3
Compromised computer	$6 - $20
Phishing website hosting (per site)	$3 - $5
Verified PayPal account with balance (balance varies)	$50 - $500
Unverified PayPal account with balance (balance varies)	$10 - $50
Skype account	$12
World of Warcraft account (one month duration)	$10

Table 1. Advertised prices of items traded on underground economy servers.

should control user passwords. The last decision to be made is how access will be granted to the users; this includes how they will provide credentials, how their credentials will be authenticated, and how to track the user's authentication from one page to another.

Before discussing how to create a secure user management system it is important to know why security at these steps should not be taken lightly. Even if a website does not contain social security numbers, addresses, employers, and other sensitive personal information, they may still have a few things in common—the same user credentials. It is important to note that many people use the same username and password combinations where possible to reduce the amount of information they have to remember. With this in mind, it is tempting for someone with malicious intent to find the username and password combination that works on a simply designed site with low security and use the same information to gather personal information (Pozadzides, 2007). For example, if a person uses the same credentials on a bank website as they do for a smaller website with low security then it would make more sense to go after that site in an attack. Once the correct username and password combination is found on the insecure site, simply try the same login information on more secure site that contains personal information. Now that you know how some thieves operate, here are some ways to defend against their attacks.

One of the most important things to do when developing a website that will house sensitive information is to keep that information secure. Store the data in an encrypted format that will help protect the user's identity. Information that does not have to be unencrypted should be stored using a hash, this is also known as one-way encryption. This means that once the data is stored, it can not be ciphered back into plaintext. Items such as passwords, secret questions, and other information that is provided by the user to confirm their identity can be stored this way. There are many different ways of creating and using hashes with user

information. Existing hashes, such as MD5, SHA-1, and the more secure SHA-512, are commonly implemented easily through most dynamic web languages. In the rare event that no hash method exists with a language, or for programmers that are good in math and just want to have fun, custom written hash methods can be created (Veness, 2005). It might not seem right to store passwords in a way that can not be deciphered; however the user's passwords will remain safe in the event that someone gains access to a list of encrypted account information. So how does the website understand whether the user entered the correct password? With one-way encryption methods, the hash is always the same assuming the same phrase is entered. Take the following example; using the SHA-1 hashing method the phrase "password" results in "5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8". Getting the original phrase from the hash would be very hard and very time consuming but when the user provides the correct password the method will result in the same hash. Even when the user makes a simple change, such as entering the phrase "Password" with a capitol "P" the result is drastically different "8be3c943b1609fffbfc51aad666d0a04adf83c9d". Simply by comparing the stored hash with the newly created hash, we see that the password is different and should not allow the user to login. This same idea can be applied to the secret questions, usernames, or other forms of identification, to keep that information safe from malicious activity if the database or other information store is viewed. All other sensitive data, such as banking information, which needs to be stored, should be encrypted using a method that allows decrypting. Any information that does not need to be gathered should be removed after using, such as a credit card number for a single transaction. This step would be a good time to create minimum and maximum length requirements for usernames. If this is in an organization, it would be a good idea to create a standard for usernames, such as making all usernames the last name then first initial of the person.

The next step is to determine a password policy. No matter how well data is encrypted or hashed, it can still be recovered using the time and resource consuming cracking method called brute force. Brute force is based on a simple principle: if it doesn't work then try again. This method uses every possible combination of characters to find the plaintext version of the encrypted data and it can take hours, days, even years, to find a password. The table shows the number of possible combinations and the approximate amount of time it would take to run a brute force attack. Obviously, the stronger the password the more time it would take to crack using this method. The best way for websites to defend against brute force attacks is to require their users to have strong passwords. This could mean requiring a combination of uppercase or lowercase letters, numbers, and symbols. Length of a password is also a very important factor, even if a password contains all of those other requirements, it still can be fast to crack if it is only a few characters. To encourage the website users to become interested in their security, give them a few suggestions to help create and remember passwords. One of the more popular recommendations is to use a phrase or lyrics and use the first letter of each word in your password, adding numbers or special characters where appropriate. It is not advised to convert letters to numbers or symbols, for example replacing the letter "a" with "@", as hackers have come to expect those (Brentnall, 2007).

Length	Digits	Upper, Lower, and Digits	Upper, Lower, Digits, and Symbols
2	100 <1s	3,844 <1s	9,216 <1s
3	1,000 <1s	238,328 <1s	884,736 <1s
4	10,000 <1s	14.7 Million 1s	85 Million 8s
5	100,000 <1s	916 Million 1m, 31s	8 Billion 13m, 35s
6	1 Million <1s	56.8 Billion 1h, 34m, 40s	782 Billion 21h, 44m, 35s
7	10 Million 1s	3.5 Trillion 4d, 1h, 50m	75 Trillion 86d, 23h, 22m
8	100 Million 10s	218 Trillion 252d, 17h, 10s	7 Quintillion 22y, 319d, 10h

Length	Upper or Lower	Upper, Lower, and Symbols
2	676 <1s	7,396 <1s
3	17,576 <1s	636,056 <1s
4	456,976 <1s	54.7 Million 5s
5	11.8 Million 1s	4.7 Billion 7m, 50s
6	309 Million 30s	404 Billion 11h, 14m, 16s
7	8 Billion 13m, 23s	34 Trillion 40d, 6h, 28m
8	208 Billion 5h, 48m, 2s	3 Quintillion 9y, 178d, 5h

Digits	123456789
Upper	ABCDEFGHIJKLM
	NOPQRSTUVWXYZ
Lower	abcdefghijklm
	nopqrstuvwxyz
Symbols	*Space*`~!@#$%^&*
	()-_=+[]{}|;:'"<>,./?

Table 2. Number of Combinations and Time required for Brute Force Attacks.

When creating password requirements, it is a good time to consider other password expiration and usage policies. The password expiration policy would cover things such as how

often should a user change their password and should the user be allowed to use the same password. Some experts suggest that requiring users to change their password too often and requiring different passwords for every change could have negative results (Lovelace, 2005). Demanding requirements for passwords can result in an overload of information causing the user to write passwords instead of remembering them. Another common password policy that should be set involves how many attempts users get at attempting to authenticate before an account becomes temporarily disabled. This helps fight against many different types of attacks, including brute force. An example of this would be after 5 unsuccessful attempts to login to the site lock the account for 30 minutes before the user is allowed to login again. This discourages brute force and dictionary attacks, an attack that uses normal words found in a dictionary as the password, by making the attacker wait a long period of time before resuming their attack. This significantly lengthens the time required to break into an account and deters most attackers.

When creating password policies for the users, it is a good idea to create a set of policies for the server usage of passwords. Some examples include never e-mailing passwords to users, never allowing users to see the passwords they have entered, and removal of passwords once the account has been removed or has expired. When passwords are e-mailed to users, they are sent insecurely over the Internet where anyone in the right place at the right time can see the password being transferred in plain text. Passwords should never be displayed anywhere on a website, if a user happens to visit the page while others are watching their password could be exposed. The password could also be captured if a user steps away from their computer in a public place while they are logged into the website. The policies above place the password outside of the control of the server, the next policies deals with how the password should be handled within the server. Eventually users will stop using their account or it will need to be

deleted for various reasons; when this happens, some account information maybe stored simply for record keeping or historical reference. There is some information that should be deleted when the account is removed; the most obvious is the user password but also the user's e-mail address and any information that could associate the user with another website. This is common in social networking sites where users can add contacts using an address book from another site. Creation of the password policies for both end users, such as the requirements, and policies for the use of personal information will help define how the interface for passwords and other user information should be stored. Having these rules in place before beginning work on the website will allow developers to reference the regulations that have already been set in place and create a website that conforms to the set standards.

Once the encryption or hash method has been determined, and the policies have been created, it is time to start working on the actual site. The developers should consider the encryption method when designing how the user information will be stored, for example if the encryption or hashing method produces an 80 character string then the database field should be created to store at least that size. There are also some hashing algorithms that require the storage of two pieces of information, sometimes referred to as the salt and hash, which creates different demands on how the information storage should be carried out. Developers should also keep in mind that users will forget their username or password and should build in a system for recovering that information. It is often acceptable to send the username through e-mail, which would require the user to remember their e-mail address. Password recovery is not as simple; passwords should never be sent through e-mail because it is often insecure. To recover a password the user should have to provide their username and some other method of authentication, such as secret questions, recent account activity, or anything unique that only the

user would know. For password recovery, using a combination of the two methods will provide additional security. Doing this could be accomplished by the site requiring that users enter their username at which time an e-mail is sent to the user that contains a unique link that expires after a few hours. Once clicking this link the user can answer secret questions and, if correct, could be prompted to change their password. The combination of these two methods would require more sophisticated techniques to gain unauthorized access to an account. Of course, there is no need for all of this security if the information is not secure while in transit. When requiring authentication or transfer of any other personal information, the website should use the highest possible encryption available. The most common encryption for secure web traffic is 128-bit or higher; many web browsers will warn the user if encryption of less than 128-bit is being used.

Now that the end user has submitted their correct username and password, the challenge of how to track the user's session begins. Tracking users does not mean that information is being collected about their browsing habits; it simply means that the same authenticated user remains authenticated while they are using the website. There are various methods of doing this, some websites use sessions others use cookies. No matter which method is used to keep track of the users, there are some simple rules to follow. The most important thing to remember is to keep any information that could be used in an attack on the server and away from the clients. This means that usernames, user IDs, account numbers, or any other personal information should not be used in cookies or session variables. A simple way to avoid this is to create a unique ID that is randomly generated when the user is authenticated and track the user based on that ID. Since no personal information, or data that could be linked to the account, is being used there should be no problem if a user forgets to logout and their cookies or session variables were accessed. Once the user logs in again, the unique tracking number is overwritten with a different

unique ID. For additional security, record the IP address and browser ID that authenticated and ensure they remain the same throughout the entire time the user is on the site. If the unique ID, IP address, or browser ID changes anytime throughout the session then it is likely that someone may be attempting to compromise the account through methods other than password cracking and the account should be logged off. For sites that require users only to be logged in for a certain amount of time, simply record the time the user authenticated and compare it to the current time, when the difference becomes too great, alert the user that their session has expired and remove the unique ID from the database, effectively logging out the user. Using this type of system can also alert the user of someone attempting to access their account because it will continuously log them out of the system. For example, if the user jsmith successfully authenticates to the server from work, a cookie could be sent to the client with a unique ID, lets say 5d6h43, and the same unique ID is stored in the database on the server. Also stored in that database is jsmith's IP address and browser identification tag. If someone were to get their hands on jsmith's unique ID and create their own cookie to try to emulate the session, the IP address and browser ID would be different and their authentication would be rejected. However if someone were to get jsmith's password and attempted to login to the system, the unique ID would change once the new user is authenticated and it would terminate the original session. When this happens, the original user knows that someone else has gotten into their account and they should change their password as soon as possible. This makes it possible to limit the sessions to only one at a time per user so if jsmith forgets to log off and attempts to access the website from home, a new unique ID would overwrite the one currently in the database and make the old unique ID useless which would automatically log off the computer at work. This type of system makes it easy for administrators to regain control of an account that has been

compromised; they simply change the account password and close a session that may be a malicious user by removing the unique ID from the database.

Whether the site is authenticating with an LDAP server, passwords stored in a database, or any other type of authentication method, a database of active sessions including the unique ID and other items mentioned earlier can be created to work in conjunction with the authentication method. When using an authentication method that is located on a different server, remember to secure the traffic between the two servers by using a secure protocol or creating a secure tunnel. When the servers are located on a controlled network that other computers can not access it might be acceptable to use insecure communications between the two servers, but it is still recommended to always transfer any authentication data securely.

Even when a website is extremely secure and very hard to break into, there will still be malicious people attempting to break in. When a certain IP address is attempting to connect to an account a large number of times, it can put strain on the web server. A simple way to reduce the strain on the server and prevent accounts from being comprised by the computer is to create a blacklist. A blacklist is a list of IP addresses, IP address blocks, or hosts that can not access the server. Most web servers, such as Microsoft Internet Information Services and Apache HTTPd, support the creation of blacklists (Host My Site 2007). Blacklists can be created at the firewall level that will prevent the traffic from ever reaching the web server, for example iptables in Linux supports the creation of blacklists (Lawrence 2005). Another blacklist option would be to write it into the website where you create a database of blacklisted IP addresses and when the address attempts to access the website return a page that says they are blocked. Another method of controlling who accesses the site is through whitelists. Whitelists are the opposite of blacklists because they are a list of addresses, blocks of addresses, or hosts that are allowed to

access the website and all others will be blocked (Hall 2007). This sort of list is useful in situations where people should only be accessing the website from certain locations, such as Intranet sites only being accessed from computers located on the same network.

Creating a set of policies, controlling who can access your server, sending and storing information in an encrypted format, and all of the other tips mentioned will help secure websites that store account information and personal data. Keeping these things in mind while developing websites will reduce the risk of data being stolen through the web interface, and will reduce the risk of the data being deciphered if the information is accessed through other methods.

References

* Brentnall, V (2007). How to pick a pA5sw0rD. *Medical Economics, 84*(8), 49-52.

* Hall, M. (2007). Get Serious. *Computerworld, 41*(12) 44.

Henry, P. (2007). *Web 2.0: New risks, new rewards*. Retrieved July 2, 2007 from http://news.zdnet.com/2100-9588_22-6190771.html.

Host My Site (2007). How do I block an IP address, through IIS, from viewing a website on my server? Retrieved July 9, 2007 from http://www.hostmysite.com/support/dedicated/IIS/blockip/.

Krebs B. (2007). *Stolen Identities Sold Cheap on the Black Market*. Retrieved July 9, 2007 from http://blog.washingtonpost.com/securityfix/2007/03/stolen_identities_two_dollars.html.

* Lassila, O., & Hendler, J. (2007). Embracing "Web 3.0". *IEEE Internet Computing 11*(3) 90-93.

Lawrence, A. (2005). Blacklist Unwanted IP Addresses. Retrieved July 9, 2007 from http://aplawrence.com/Words2005/2005_05_01.html.

* Lovelace, H (2005). Password Complexity Puts Security At Risk. *InformationWeek* (1027) 68.

Veness, C. (2005). *JavaScript Implementation SHA-1 Cryptographic Hash Algorithm*. Retrieved July 1, 2007 from http://www.movable-type.co.uk/scripts/sha1.html.

Privacy Rights Clearinghouse (2007). *How Many Identity Theft Victims Are There? What IS the Impact on Victims?* Retrieved June 30, 2007 from http://www.privacyrights.org/ar/idtheftsurveys.htm.

Pozadzides, J. (2007). *How I'd Hack Your Weak Passwords*. Retrieved June 26, 2007 from http://onemansblog.com/2007/03/26/how-id-hack-your-weak-passwords/.