Preventing a Brute Force or Dictionary Attack: How to Keep the Brutes Away from Your Loot

By Bryan Sullivan, SPI Dynamics

To understand and then combat a brute force attack, also known as a dictionary attack, we must start by understanding why it might be an appealing tool for a hacker. To a hacker, anything that must be kept under lock and key is probably worth stealing. If your Web site (or a portion of it) requires a user to login and be authenticated, then the odds are good that a hacker has tried to break into it. In terms of processing power, it is expensive for a Web site to require authentication, so it is usually only required when the site stores valuable private information. Corporate intranet sites can contain confidential data such as project plans and customer lists. E-commerce sites often store users' email addresses and credit card numbers. Bypassing or evading authentication in order to steal this data is clearly high on a hacker's priority list, and today's hackers have a large library of authentication evasion techniques at their disposal.

Session hijacking attacks such as Cross-site Scripting can steal a user's authentication token and transmit it to a malicious third party, who can then use it to impersonate the legitimate user. SQL injection attacks can also be very effective at bypassing authentication. By sending a specially-formatted username and password combination containing SQL code to the login form, an attacker can often trick the server into granting him unauthorized access. These types of attacks get a lot of attention since they are creative, elegant, and effective. However, there is another type of attack that can be just as effective, if not as elegant or creative. A brute force attack (or dictionary attack) can still be a dangerous threat to your Web site unless proper precautions are taken.

The brute force attack is about as uncomplicated and low-tech as Web application hacking gets. The attacker simply guesses username and password combinations until he finds one that works. It may seem like a brute force or dictionary attack is unlikely to ever succeed. After all, what are the odds of someone randomly guessing a valid username and password combination? Surprisingly, the odds for a brute force attack can be quite good if the site is not properly configured. There are several factors that work to the hacker's advantage, the most important of which is human laziness.

Don't Be Lazy – Choose a Password Carefully!

Generally, people do not remember complicated passwords very well. If users are allowed to create their own passwords, they will often create very simple ones like "password", "1234", their spouse's name, or their favorite sports team. Passwords like these are easy for the user to remember, but unfortunately they are also easy for someone else to guess. Furthermore, any serious hacker who attempts a brute force attack will not be sitting at a Web browser, guessing at authentication credentials and typing them in. He will be using an automated tool for the brute force attack that can make thousands of requests per minute with credentials generated from a large list of possible values. Often this list is an actual dictionary, hence the term "dictionary attack." If a user chooses a

common password, such as a dictionary word, the automated tool will eventually guess it, and the user's account will be compromised.

Also, once the brute force attack has revealed a valid username and password combination for one Web site, the hacker knows that the same combination is likely to work for other Web sites. In a study conducted by the University of Wichita, more than half of the test subjects reported using the exact same password for multiple sites. This laziness works to the hacker's advantage. If, for example, a hacker is able to use a dictionary attack to obtain a valid user credential for Amazon.com, then it is probable that the same credential would be valid for other popular Web sites, such as eBay.

Sidestepping a Dictionary Attack with Username Selection

Of course, a password is only half of the required login credential. A username is also required. While it is less likely that a dictionary word would be used as a username, there are still some common usernames that hackers are certain to try with a brute force attack. First among these are "admin" and "administrator". These names are especially dangerous since they are not only easily guessed, but the accounts they represent are usually highly privileged administrative accounts. If the hacker's dictionary attack could gain access to an administrative account, he could probably do much more damage to the system than he could if he gained access to a regular user's account.

Administrative accounts are not the only problem: many Web applications and Web application frameworks create default users during installation. If the site administrator does not remove these default users or at least change their passwords, these accounts will be easy targets for a dictionary attack. Finally, when users are allowed to choose their own usernames, they often choose their email address, since it is easy to remember. Once again, the user's laziness is a benefit to a hacker using a brute force attack. Armed with a list of email addresses (perhaps obtained from a spammer) and a dictionary of passwords (easily obtained anywhere), an attacker has an excellent chance of breaking into at least one user's account.

Countering a Brute Force Attack with a Strong Password Policy

The primary defense against a brute force attack must be enforcement of a strong password policy. As mentioned earlier, dictionary words make poor passwords. Password size is also important: the longer the password, the more difficult it will be to force. While there is no strict definition of a strong password that will be harder to determine via a dictionary attack, some good guidelines would be:

- Minimum length of at least seven characters
- Must include both upper and lower case characters
- Must include numeric characters
- Must include punctuation

These guidelines may seem overly strict, but there is little chance that a password created with these restrictions will be found with a brute force attack. There are almost *70 trillion* combinations of characters that can be seven digits long and can include upper case

characters, lower case characters, numbers, and punctuation. Even a dictionary attack tool that could make one hundred requests per second would still take over 11,000 years before it would be statistically likely to guess the password.

Obviously, most Web sites will want to block a dictionary attack much sooner than 11,000 years into the attack. Many organizations use an intrusion detection system (IDS) to detect an abnormally high number of requests coming from a single user. This is a good idea, but it is not sufficient to prevent the brute force attack. A clever hacker will simply reduce the bandwidth used by his automated tool until it falls under the alert threshold of the IDS.

Other Defensive Strategies – And Why They Don't Work

Another common defense strategy against a dictionary attack is to automatically disable an account after a certain number of failed login attempts. For example, if the server detects that the user "bobsmith" has provided an incorrect password three times since his last login, the server might decide that the "bobsmith" account is the subject of a brute force attack and will disable it. The account may automatically reactivate after 30 minutes, or the user might have to contact the site administrator to have the account reactivated. In either case, automatically disabling user accounts is a poor security mechanism to fight a dictionary attack. In the first place, by disabling accounts the system has traded an authentication evasion vulnerability for a denial of service vulnerability. If an attacker can disable an account by incorrectly guessing its password three times every 30 minutes, he can effectively prevent that user from ever accessing the system. Imagine how damaging a dictionary attack could be if it were used against an administrative account.

In the second place, locking out accounts is ineffective against a brute force attack because this technique assumes that the attacker is keeping the username constant and varying the password. What if the attacker instead kept the password constant and varied the username? We already know that a large percentage of users use common passwords like "password". A hacker using a dictionary attack could try "password" for each of the users in his username list, which would not only have a high chance of success, but would also evade the account lockout logic. An attacker could make thousands of login attempts, and even if every one of them failed, the system will only register one incorrect login per account.

A Better Defense: Incremental Delay

A better strategy for blocking any brute force attack is to incrementally delay the page response after failed login attempts. After the first failed login attempt, for example, the response would be delayed by one second. After the second failed attempt, the response would be delayed by two seconds, and so on. A one-, two-, or even six-second delay is probably not going to bother a human user too seriously. Certainly he will find it less irritating than having to wait 30 minutes for his account to reactivate because he accidentally left his caps lock key on. On the other hand, an incrementing delay can

completely defeat an automated tool being used for a brute force attack. Assuming the tool could normally make ten requests per second, the time it would take to make one thousand requests would jump from two minutes to five days. This pretty much renders the brute force attack tool useless. An incrementing delay also solves the problem of the attacker holding the password constant and varying the username. Since the system tracks failed login attempts on a user session basis and not an authentication credential basis, the delay logic cannot be bypassed this way.

There is one serious shortcoming to the incrementing delay approach: state must be kept in order to record the number of failed login attempts by the current user. The dictionary attack tool can be set up to begin a new session on every request by never sending a session identification token to the server. In this situation, the server will not be able to track the number of failed logins, and the delay will not be properly applied. It is possible to track a user from his IP address instead of his session token, but this technique has problems as well. Sometimes multiple users share a single IP address, and sometimes a single user can change IP addresses between requests. While the incrementing delay technique is not perfect, in many cases it is a better solution to fighting a dictionary attack than the widely used practice of locking out accounts after failed login attempts.

Carefully Word Your Error Messages

Finally, it is important to create appropriate error messages in response to failed login attempts. Many Web sites inadvertently aid hackers by providing overly helpful error messages. Consider the difference between the messages “User ID not found” and “Incorrect password.” These messages give a lot of information to a potential attacker. “User ID not found” tells the hacker that the user he is trying to determine via brute force attack does not exist in the system. There is no point in continuing to try different passwords for this username. He can continue on to the next username in the list, saving himself thousands of useless requests and hours of time. On the other hand, “Incorrect password” tells him that the username he has tried with his dictionary attack does exist, but that the password is wrong. Now he knows that he has a potential victim and can focus his efforts on breaking that user’s password. It is much safer for the application to respond with an ambiguous message like “Incorrect username or password” when a login attempt fails. There is no way to tell from this error which part of the credential was invalid. Therefore, there are no clues that a hacker can obtain from this error that can help him reduce his workload and break the system faster.

Conclusion

In conclusion, sometimes old, boring attacks can work just as well as the new, exciting ones. Low-tech as it might be, a brute force attack can be very effective at compromising your Web application unless proper defenses are used. The first and foremost method of defeating a brute force attack is to require all users to choose a strong password. Passwords should be required to contain at least seven characters, with mixed upper- and lower-case letters, numbers, and punctuation. Also, consider implementing an incrementing response delay routine in your application in place of an automatic account

lockout. Finally, be sure to display nondescript, ambiguous login failure messages such as “Invalid username or password.” Messages like this provide no extra information about the system that a hacker using a dictionary attack can take advantage of to lighten his workload. Following these guidelines will help you protect your application and your users from the brutes of the world.

About the Author

Bryan Sullivan is a development manager at SPI Dynamics, a Web application security products company. Bryan manages the DevInspect and QAInspect Web security products, which help programmers maintain application security throughout the development and testing process. He has a bachelor’s degree in mathematics from Georgia Tech and 11 years of experience in the information technology industry. Bryan is currently coauthoring a book with noted security expert Billy Hoffman on Ajax security, which will be published in summer 2007 by Addison-Wesley.