Entrust Solutions Nepal

Account locking scheme: why is it not “alone dependable” measure against brute force attacks?

Abhibandu Kafle
Entrust solutions Nepal

Background:

Everyone knows what a brute force is; it is a series of attempts made to access an account of a legitimate user by an attacker. It doesn't necessarily mean trying everything possible, rather so called hybrid-brute force attack [brute force based on personal guess or most widely used passwords] are used in dictionary to make attempts.

However, several countermeasures have already been developed and deployed against this kind of attack. Beforehand, strict password policy is also a sound mechanism to make it harder for an attacker.

Current trends:

It is needless to say, the brute notions that account locking schemes are measures to stop brute force attacks still exist.
Considering account lockout solution as only existing solution, different possible measures exists.
Sample implementation in php:

1) Initialize session [session_start()]

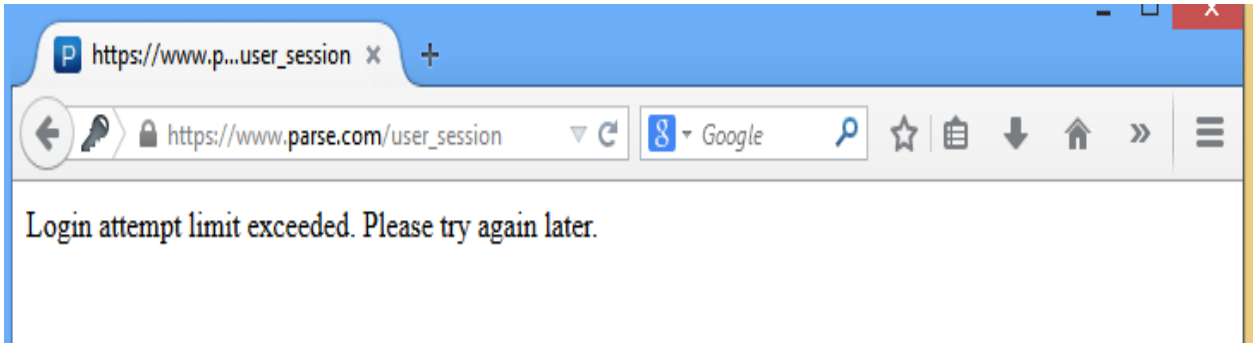
2) Define a session counter
   [if (!$_Session['count']){
   $_Session['count'] = 0;}]

3) Increase counter for every login attempts

4) Check if count has exceeded max_allowed_attempts
   [if ($_Session[count]>$max_allowed_attempts){
   Die('Login limit exceeded. Please try again later.');
   }]

A Paradigm in parse.com:

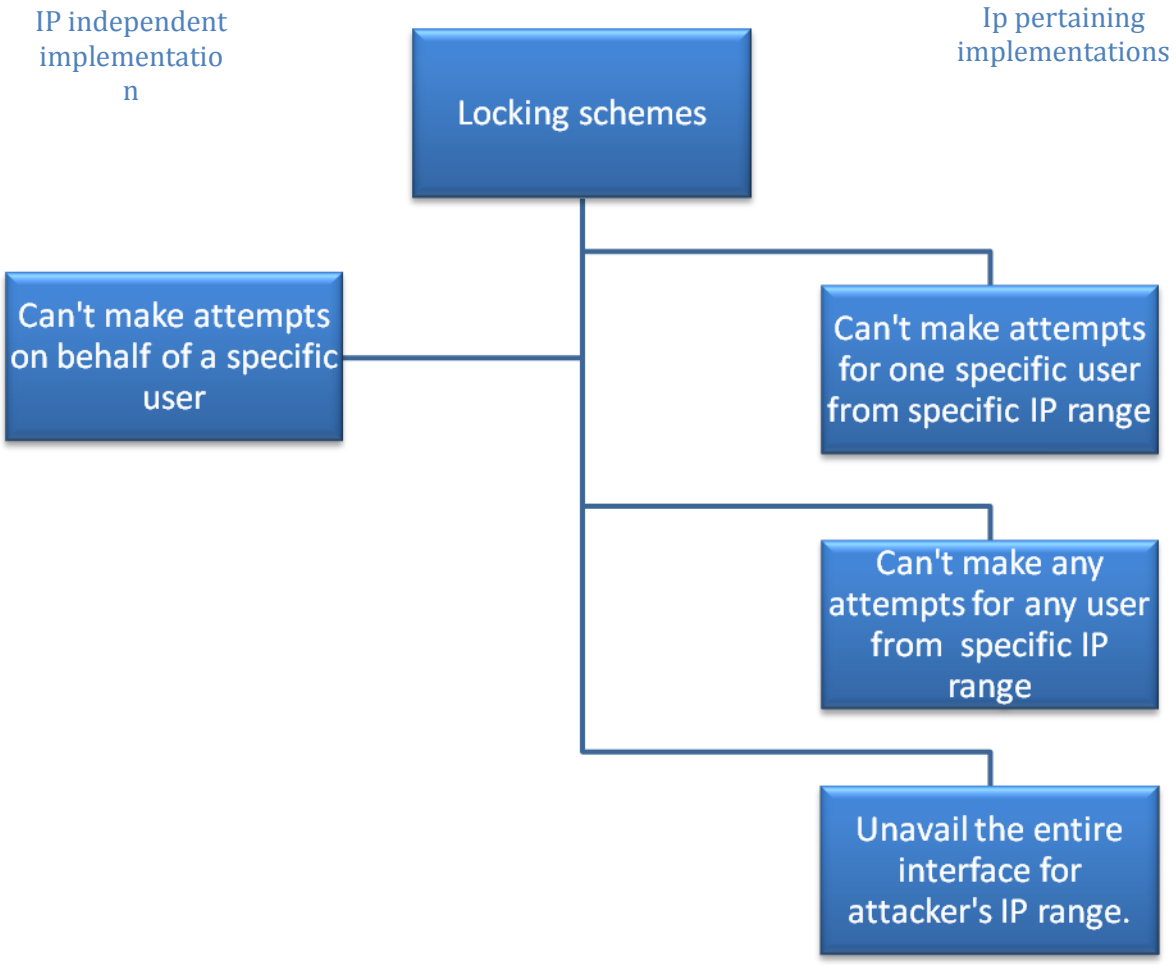
The exact behavior was seen (now fixed) in parse.com, one of facebook's acquisitions, which also tracked the user being brute-forced, regardless of IP that was origin of attack. And after 100 attempts, application locked out the user even if he was logged in from other browser.



Facebook itself also uses similar concept to prevent brute-forcing of certain forms, but instead of session, it checks cookie variables.
Tip: You can use facebook's sign up form to brute force any user account, If you could clear cookies every 25 requests.

So, its plain truth that even giants of internet still use account locking scheme as a countermeasure against brute force attacks, account locking scheme's widespread usage is evident.

Some popular implementation schemes



Why none of above resolves the problem?

Let's see failure of popular used IP locking schemes:

1) IP independent implementation :
   If a specific username is locked out for some timeframe, independent of IP responsible for brute force attacks:

   i) If an attacker would like to disallow a user to login to portal for specific amount of time, he is free to do it anytime. All he has to do is make false login attempts on behalf of a username.

   ii) If an attacker has inimical notion, he'd enumerate the users, and make exactly the no. of attempts required to lock the user's account. This will make the portal useless because no users can login, because they are locked.

   iii) An attacker could continuously lock out same account, effectively disabling the account.

   Scenario: Suppose this scheme is implemented in e-bidding portals. An attacker places his bid and locks every other account until you can no longer place a bid.

2) IP specific implementation
   If a particular IP range, responsible for brute forcing, is being blocked rather than a particular username, this is a little safer side. However, it has limitations, as well:

   i) Using tor proxies and targeting the same user makes the measures against brute-force feeble.

   ii) Blocking a specific network range doesn't always means blocking a single user. There can be prominent no. of users relying in those networks [like public wifi, college network, workplace internet]
   Even some ISP's implement NATing(Network address Translation) techniques and distribute a common range of public IP to its users. Brute force from one IP address means the portal is inaccessible to any of users in that range.

Some scenario where locking can be good idea:

If you own a site, and don't have too much resource to spare, supernumerary login attempts aren't desirable.

If you think, lockout for few moments won't affect User experience in your application.

Solution:

Sometimes IP-specific implementation can also be a mediocre solution: attack surface is reduced from "persistent attacker anywhere in the world" down to "persistent attacker on the same IP" -- a far, far narrowed attack surface, unlike the condition, when IP is not taken into account, a persistent attacker could keep a user locked out by continually trying to brute force from anywhere in the world.

This could be something like

```
function humanRequest(){
    // decide if the request lag is humanistic or bot speed
    //maybe estimate the time lag between two consecutive requests
}

if(!humanRequest()){
    // redirect to a Captcha page or die with a warning.
}
$uid = getUsername($username);
if($uid > 0){
    // validated request
}
else{
    // increase attempts counter
    // you could have a separate column for Specific IP requests
    // lock out username after counter> max_allowed_attempts.
}
```

IP-specific implementation, accompanied by Captcha, makes a perfectly compounded solution.

Conclusion:

- Every portal that has a user login panel can be susceptible to brute force. Account lockout scheme is widely used to prevent such attacks.
- Account lockout implementations are generally IP-pertinent and in some cases IP-independent.
- IP-independent username locking, in general, is not a sound way to prevent brute force, as it can lead to DOSing of all user accounts in the portal from a persistent attacker.
- IP aware locking schemes also can cause trouble when ISPs use NATing or while accessing account from public networks. An attacker can also shuffle through IP addresses and continuously brute-force the username
- A good way to prevent such kind of attacks is to implement IP-specific locking scheme with Captcha.