SMB Login Bruteforce: Intelligent PowerShell Hacking Tool

By Jason Lang

Published: 2026-02-23 · Archived: 2026-04-05 20:07:25 UTC

Intro

One of my favorite post-ex metasploit modules is smb_login. It's great for running a quick test using credentials you've discovered. One of the problems with it is that there is nothing that prevents you from locking out accounts. Plus, you have to create user list which means dumping users | cut | sed | awk, blah blah blah. (**Update**: Thanks to @ztgrace for bringing me up to speed on the MaxGuessesPerUser advanced property of smb_login.)

I wanted something that did all this work for me that would just take a password list and be smart enough to handle all the logistics, plus I wanted it in PowerShell for obvious reasons (PowerShell > Ruby. <trollygrin>). Seriously though, I'm not a Ruby dev, and every time I try, I reach gem dependency rage-level 11 in about 15 minutes. So…

- Admin Rights Required: NO
- PowerShell Version Required: 2+

The autobrute script has a few features built in making it handy for the pentester who needs creds & is short on time.

Parameters

The script receives the following parameters:

UserList

A simple text file that exists on target containing users to brute (one per line). If no list is passed, query the domain for a list of users whose badPwdCount attribute is two less than the domain account lockout threshold. Wrap paths in double quotes.

PasswordList (Required)

A comma separated list of passwords to try. Even if the Lockout Threshold is 3 attempts, pass in 10 passwords or so. The script will grab safe users to brute every password run. Wrap list in double quotes.

LockoutThreshold (Required)

The lockout threshold of the domain. Run "net accounts" on the target, grab the Lockout Threshold value and use that.

Delay

The number of milliseconds to wait between each attempt. Handy if your connection is slow, otherwise you could get odd errors. Default 100.

ShowVerbose

By default, only successes are shown. Specifying this switch will show all skipped and failed attempts. Lots of information will hit the screen. You've been warned.

StopOnSuccess

What you'd expect. After the first successful authentication, exit the script.

Workflow

The general order of the script is as follows. Assume no UserLIst was passed and the LockoutThreshold was set to 5.

1. Perform prereq checks. Be sure you can locate the PDC, etc
2. For each password in the password list, perform the following:
    3. Retrieve a list of enabled users from the domain (PDC specifically) whose badPwdCount attribute is <=3. The reason for this is is that we want all users who could not be locked out during this attempt. It is possible that a user could fat finger a password during the brute, locking their account. Unlikely, but possible.
    4. For each user retrieved:
        1. Check their badPwdCount attribute against all DCs and use the highest value. The reason this is done is the badPwdCount attribute is not replicated (source). If the highest value is greater than one less than the lockout threshold, do not test the account.
        2. If the account is safe to test, test the password against the PDC. If successful add the user to a valid users list (and never test this user again). Throw the result to the screen.

Risks: Regardless of the safety checks built in to the script, it is possible that lockouts could still occur. Replication problems between DCs, a DC that is being rebooted during processing, users who are trying as fast as they can with bad passwords, all could cause lockouts. **Always best to test before you run against your target**! It's been tested in my lab against thousands of users, but that's it. We are not liable for your slow env or the accounts you lock! 🙂

Quick Screenshot:

Running as a low priv account.

Note: If you get LDAP server unavailable errors, you might be bruting too fast. Try setting the Delay param to 500 or so.

Defense

As with any brute force attack, your logs (specifically the PDC Security Event log) will be filled with failures. This script will actually load up the logs of all DCs as each user is checked against each DC for their badPwdCount attribute. We always recommend to our customers that they be setting thresholds on alerts so that if X events fire in X seconds, you are alerted. The security event log will contain the source IP of the authentication attempt. That's your compromised machine.

Really sneaky pentesters could set the Delay to 1000+ and just let it run overnight. </evilgrin>

Comments are welcome, but please use github for any questions/bugs. Our scripts repo is here.

@curi0usJack

PS – An Empire pull request has been submitted. Keep a look out for situational_awareness/network/smbautobrute. 🙂
