AWS Security Showdown: Security Group vs NACL

A Practical Guide to Layered Defense

Narasimha Potturi
DevOps and Cloud Engineer

🛡️ Why You Need Both

Securing your AWS infrastructure isn't about one single tool. It's about defense-in-depth.

Two of the most fundamental tools for network security are Security Groups (SGs) and Network Access Control Lists (NACLs).

They both act as firewalls, but they operate at different levels and in different ways.

Misunderstanding them can leave you vulnerable... or locked out of your own systems!

Let's break down the difference.

Deep Dive: Security Groups (SG)

🧑✈️ The Instance Bouncer: Security Groups

A Security Group (SG) is a virtual firewall that controls traffic at the instance level.

Scope: Associated with one or more EC2 instances.

Rules: ALLOW rules only. You cannot create "deny" rules. By default, all inbound traffic is denied.

Key Feature: “*STATEFUL”*. This is the most important concept!

What "Stateful" means:

If you allow inbound traffic on port 8000, the SG automatically "remembers" that connection and allows the outbound (return) traffic, even without an explicit outbound rule.

Deep Dive: Network ACLs (NACL)

🛂 The Subnet Checkpoint: Network ACLs

A Network Access Control List (NACL) is a firewall that controls traffic at the subnet level. It's the first line of defense before traffic even reaches your instance.

Scope: Associated with one or more subnets.

Rules: ALLOW and DENY rules. Rules are evaluated in numerical order (lowest number first).

Key Feature: *"STATELESS"*

What "Stateless" means:

The NACL does not remember connections. If you allow inbound traffic on port 8000, you must explicitly create a separate outbound rule to allow the return traffic (on ephemeral ports 1024-65535).

The Cheat Sheet: SG vs. NACL

Comparison: At a Glance

Feature	**Security Group (SG)**	**Network ACL (NACL)**
Scope	Instance Level	Subnet Level
Type	Stateful (Remembers connections)	Stateless (Forgets connections)
Rules	Allow rules only	Allow and Deny rules
Default	Denies all inbound. Allows all outbound.	Allows all inbound & outbound.
Evaluation	All rules are evaluated	Rules evaluated in number order
Simple Analogy	Bouncer at the VIP room door	Security checkpoint at the building entrance

🧑‍💻Hands-On: The Default Behavior

Let's see this in action!

I launched an EC2 instance in a VPC.

I SSH'd into the instance and ran a simple Python web server:

```
python3 -m http.server 8000
```

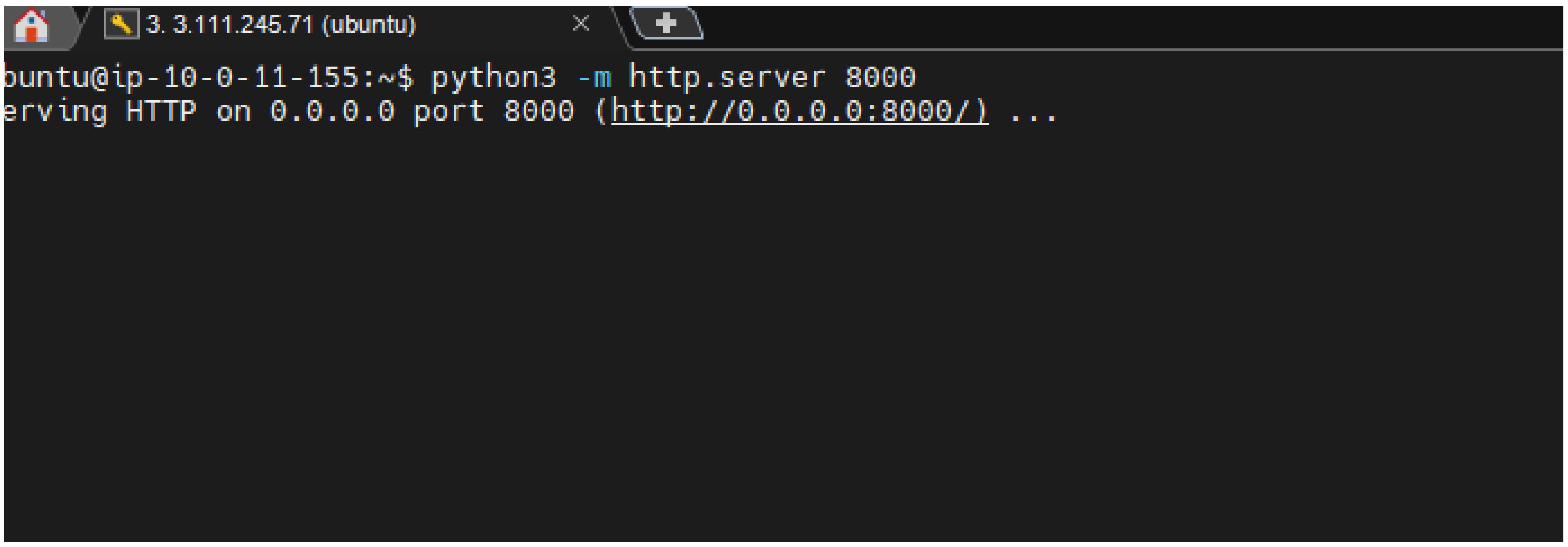
I tried accessing the instance's Public IP in my browser:

http://[Public_IP]:8000

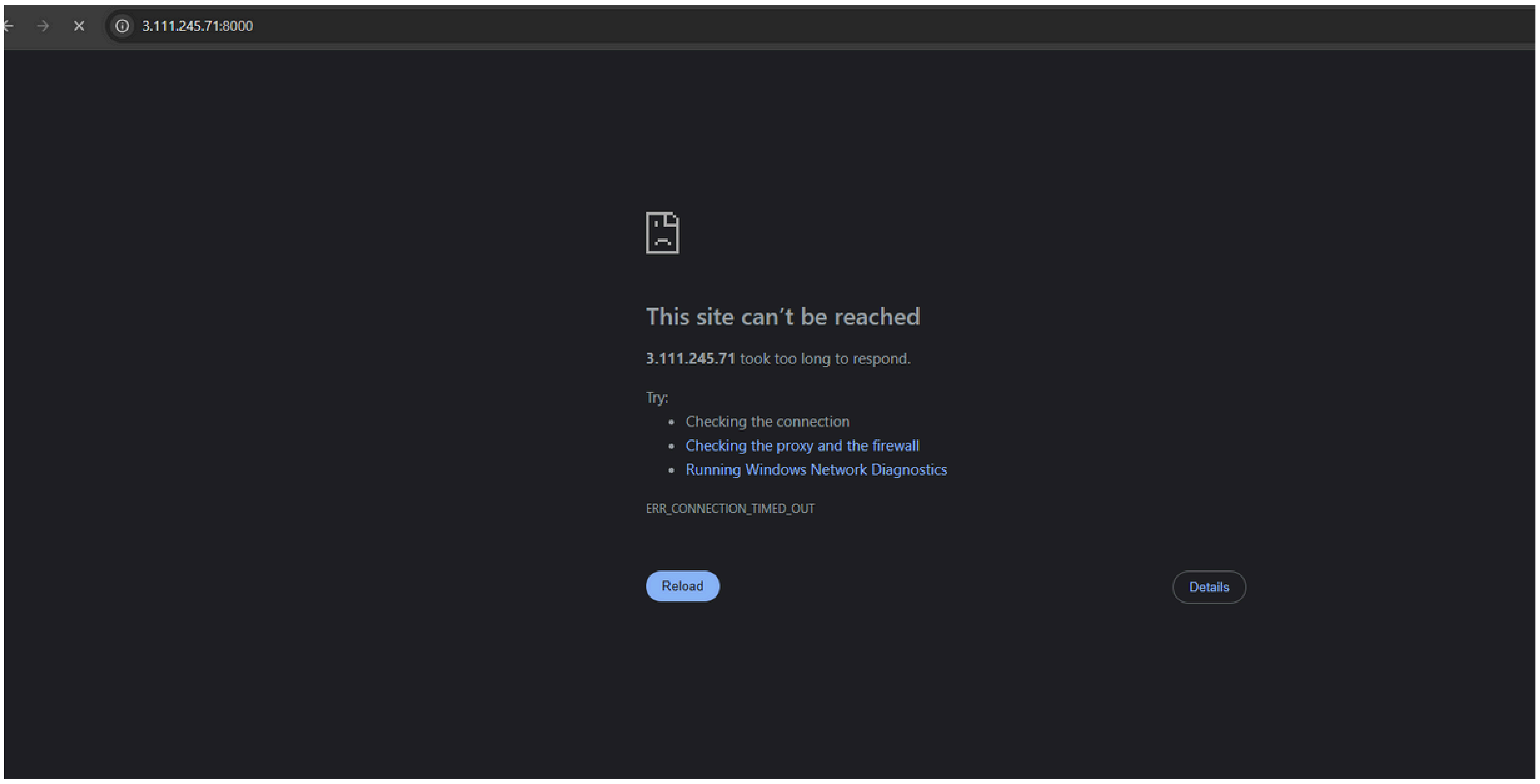
Result: Connection Timed Out.

Why?

The default Security Group denies all inbound traffic. The bouncer (SG) didn't have port 8000 on the guest list, so my request was rejected.



```
3. 3.111.245.71 (ubuntu)
buntu@ip-10-0-11-155:~$ python3 -m http.server 8000
erving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



3.111.245.71:8000

This site can't be reached

3.111.245.71 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT

👨‍💻Hands-On: The Security Group Fix

Now, let's "fix" this by telling the instance's bouncer to let us in.

I went to the Security Group attached to my instance.

I added a new Inbound Rule:

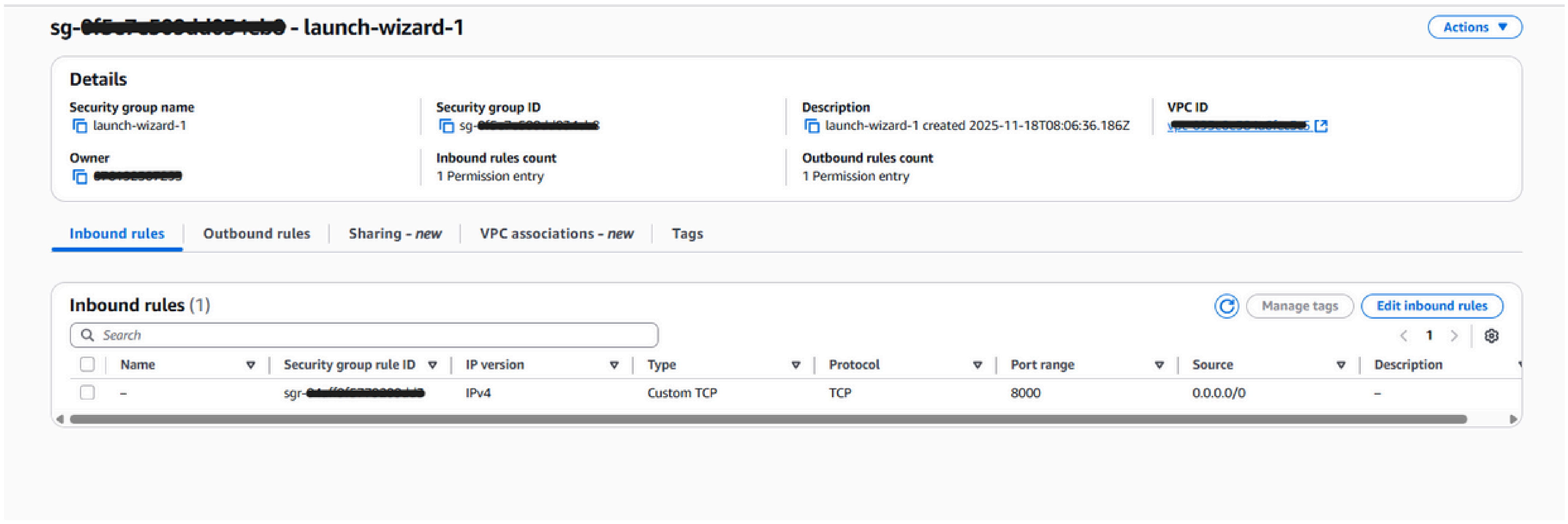
- Type: Custom TCP
- Port Range: 8000
- Source: 0.0.0.0/0 (For demo. Use "My IP" for better security!)

I refreshed my browser.

Result: Success! The Python server's "Directory listing" page loaded.

Why?

The Stateful SG now allows traffic on port 8000. It sees my request, allows it, and automatically allows the return traffic back to my browser.



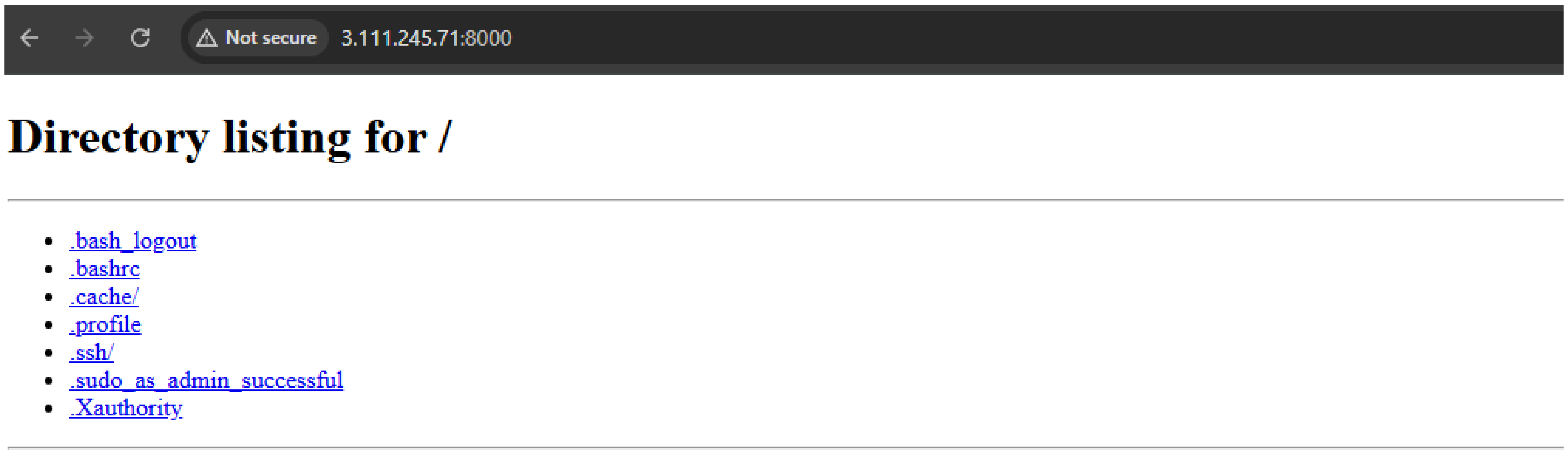
sg-[redacted] - launch-wizard-1

Details

Security group name: launch-wizard-1	Security group ID: sg-[redacted]	Description: launch-wizard-1 created 2025-11-18T08:06:36.186Z	VPC ID: [redacted]
Owner: [redacted]	Inbound rules count: 1 Permission entry	Outbound rules count: 1 Permission entry	

Inbound rules (1)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-[redacted]	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	–



Not secure 3.111.245.71:8000

Directory listing for /

- .bash_logout
- .bashrc
- .cache/
- .profile
- .ssh/
- .sudo_as_admin_successful
- .Xauthority

⚠️ **NOTE :** For this practical demonstration, I have allowed traffic from 0.0.0.0/0 (all IPs). In a real-world **production environment**, you must **restrict traffic** to only the required IP addresses or CIDR ranges, aligning with your organization's security and compliance standards. **Be careful and allow traffic only where necessary!**

👨‍💻 Hands-On: Layered Security with NACL

Our app works. But what if a DevOps engineer needs to block this port for the entire subnet as a security measure?

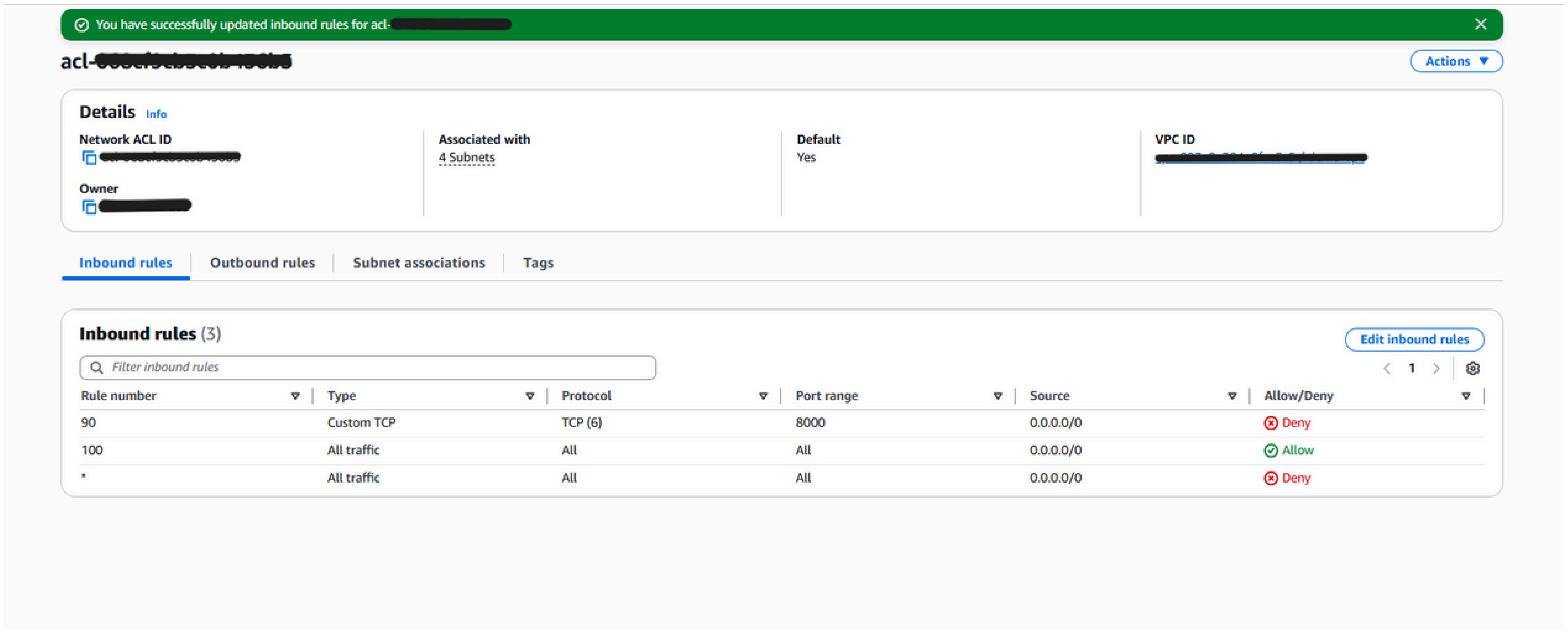
I went to the Network ACL (NACL) associated with my instance's subnet.

I added a new Inbound Rule:

- Rule #: 90 (must be a lower number than the default '100' allow)
- Type: Custom TCP
- Port Range: 8000
- Source: 0.0.0.0/0
- Action: DENY

I refreshed my browser again.

Result: Connection Timed Out!


You have successfully updated inbound rules for acl-[redacted]

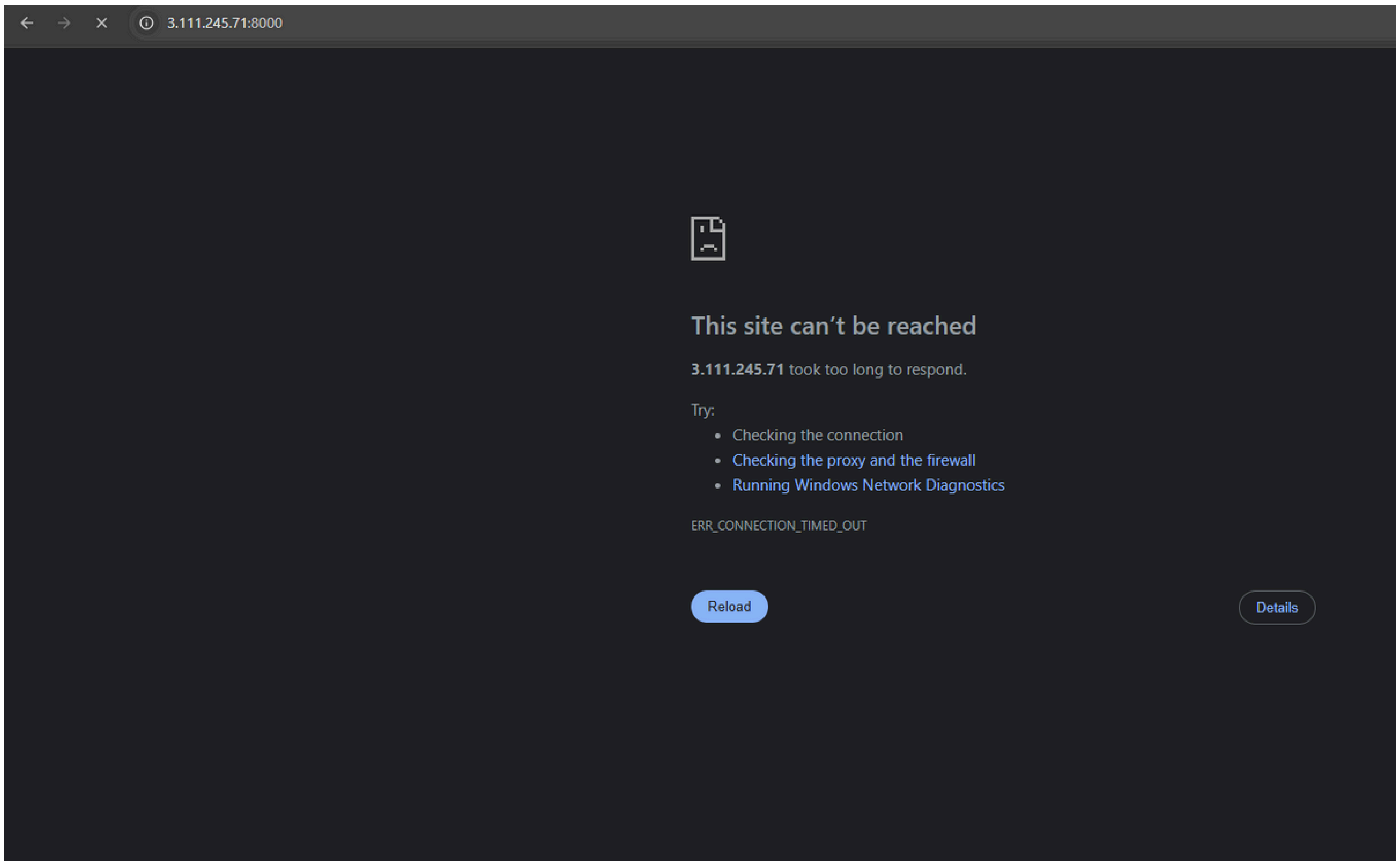
acl-[redacted]

Details Info

Network ACL ID: [redacted]
Associated with: 4 Subnets
Default: Yes
VPC ID: [redacted]
Owner: [redacted]

Inbound rules (3)

Rule number	Type	Protocol	Port range	Source	Allow/Deny
90	Custom TCP	TCP (6)	8000	0.0.0.0/0	Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny


3.111.245.71:8000

This site can't be reached

3.111.245.71 took too long to respond.

Try:
- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT

💡 The "Why?" Moment: Traffic Flow

"Wait... my Security Group allows port 8000, but it still failed. Why?"

Because of the order of operations! Traffic always hits the NACL first.

Traffic Flow:

Internet -> NACL (Subnet) -> Security Group (Instance) -> EC2 Instance

Our NACL (the building security) had a DENY rule. It blocked my request before it ever got to the Security Group (the bouncer).

This is a perfect example of "defense-in-depth." The NACL acts as a broad, stateless filter, while the SG provides specific, stateful protection.

🚀 Key Takeaways

- **Security Groups (SG)** are **Stateful** and operate at the **Instance** level. (Your bouncer).
- **Network ACLs (NACL)** are **Stateless** and operate at the **Subnet** level. (Your building security).
- Traffic is evaluated by the **NACL first**, then the **SG**.
- Use NACLs for broad rules (e.g., blocking a known malicious IP for the entire subnet) and SGs for specific rules (e.g., allowing port 22 only from your IP to a specific instance).

(Call to Action): Thanks for reading! How do you use SGs and NACLs to lock down your applications?

Follow
for more
great tips

Narasimha Potturi
DevOps and Cloud Engineer