CS 455 – Computer Security Fundamentals

Dr. Chen-Yeou (Charles) Yu

Computer Security Fundamentals

- ~~Amazon EC2 instances with Security Groups and Session Manager~~
- AWS Network Access Control List (NACL or Network ACL)
- AWS Web Application Firewall (WAF) (TBD)
  - **AWS WAF Bot Control Explained and Demonstrated**

AWS Network Access Control List

- Let’s learn some defensive mechanisms. Shall we?
- We have been hackers for a long period of time. Enough!! ☺
- First thing of all, someone may ask me what’s the difference between “NACL” and “Security Group”, we have learned that earlier, isn’t it?
- Check the next page for details (the AWS Network Firewall, Developer’s Guide)

AWS Network Access Control List

- “Stateless” rule engine which is for “NACL”. However “stateful” rule engine is for “Security Group”
- Yes, we are talking about the settings in the firewalls!
- Stateless rules are aiming for best performance (We don’t care about traffic direction or the traffic is from an existing connection)
- Wait!? Rules for what? Firewall’s rule!
  - Each rule has a rule number in AWS
  - Rules for ‘allow’ or ‘deny’
  - This can be applied onto the connection source. i.e. Hacker’s machine
- We can create a new rule or delete a rule
- So, the moment it matches the rule, it drops the traffic!

The stateless and stateful rules inspection engines operate in different ways:

- **Stateless rules engine** – Inspects each packet in isolation, without regard to factors such as the direction of traffic, or whether the packet is part of an existing, approved connection. This engine prioritizes the speed of evaluation. It takes rules with standard 5-tuple connection criteria. The engine processes your rules in the order that you prioritize them and stops processing when it finds a match.

  Network Firewall stateless rules are similar in behavior and use to Amazon VPC network access control lists (ACLs).

AWS Network Access Control List

- Stateful rule engine in this case, it allows you more complex rules.
- But one key thing to take note of is that, the “Security Groups” do not sponsor the “deny” such a kind of activity.
- On the other hand, NACL provide the function for you to run a deny against, for example, an IP address

Stateful rules engine – Inspects packets in the context of their traffic flow, allows you to use more complex rules, and allows you to log network traffic and to log Network Firewall firewall alerts on traffic. Stateful rules consider traffic direction. The stateful rules engine might delay packet delivery in order to group packets for inspection. By default, the stateful rules engine processes your rules in the order of their action setting, with pass rules processed first, then drop, then alert. The engine stops processing when it finds a match.

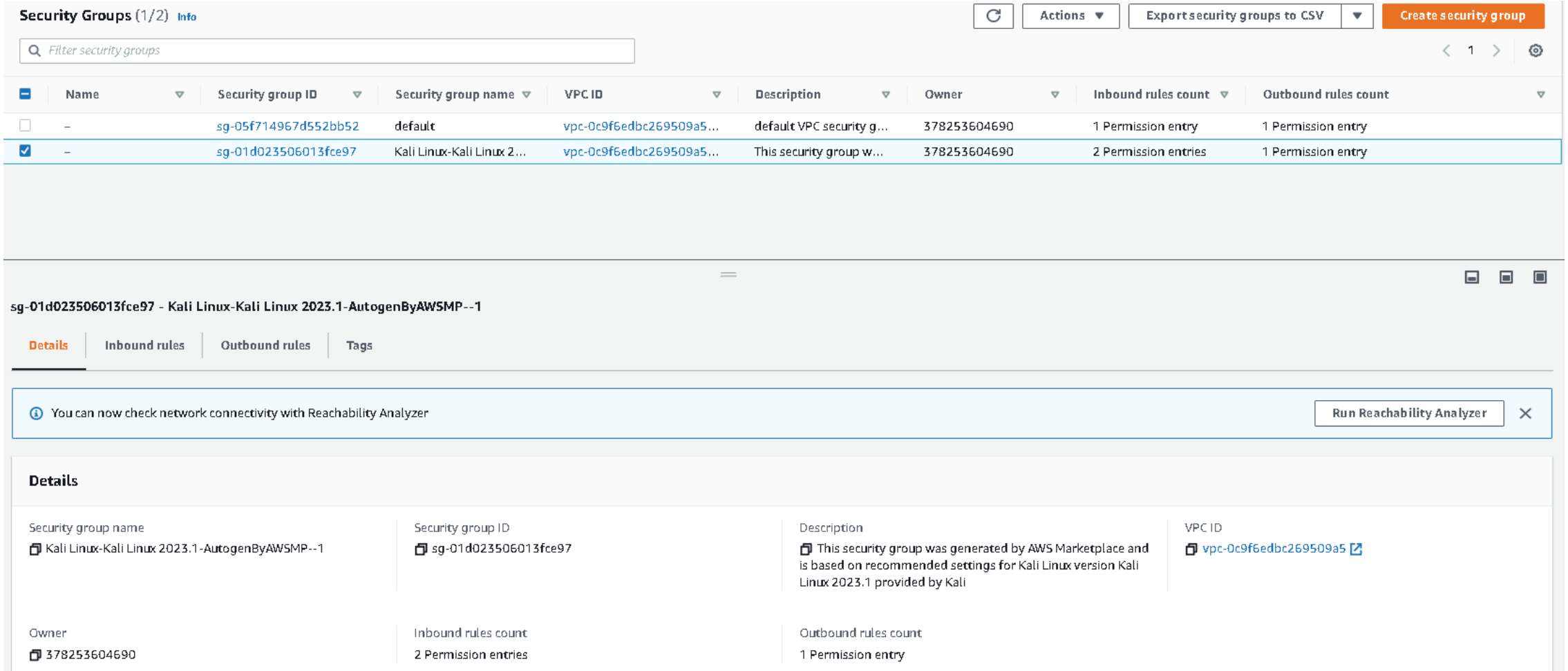
The stateful engine takes rules that are compatible with Suricata, an open source intrusion prevention system (IPS). Suricata provides a standard rule-based language for stateful network traffic inspection. For more information about Suricata, see Stateful rule groups in AWS Network Firewall and the Suricata website.

Network Firewall stateful rules are similar in behavior and use to Amazon VPC security groups. By default, the stateful rules engine allows traffic to pass, while the security groups default is to deny traffic.

Stateful rule engine considers traffic direction!

AWS Network Access Control List

- Previously, we introduced the “security group” for inbound / outbound rules

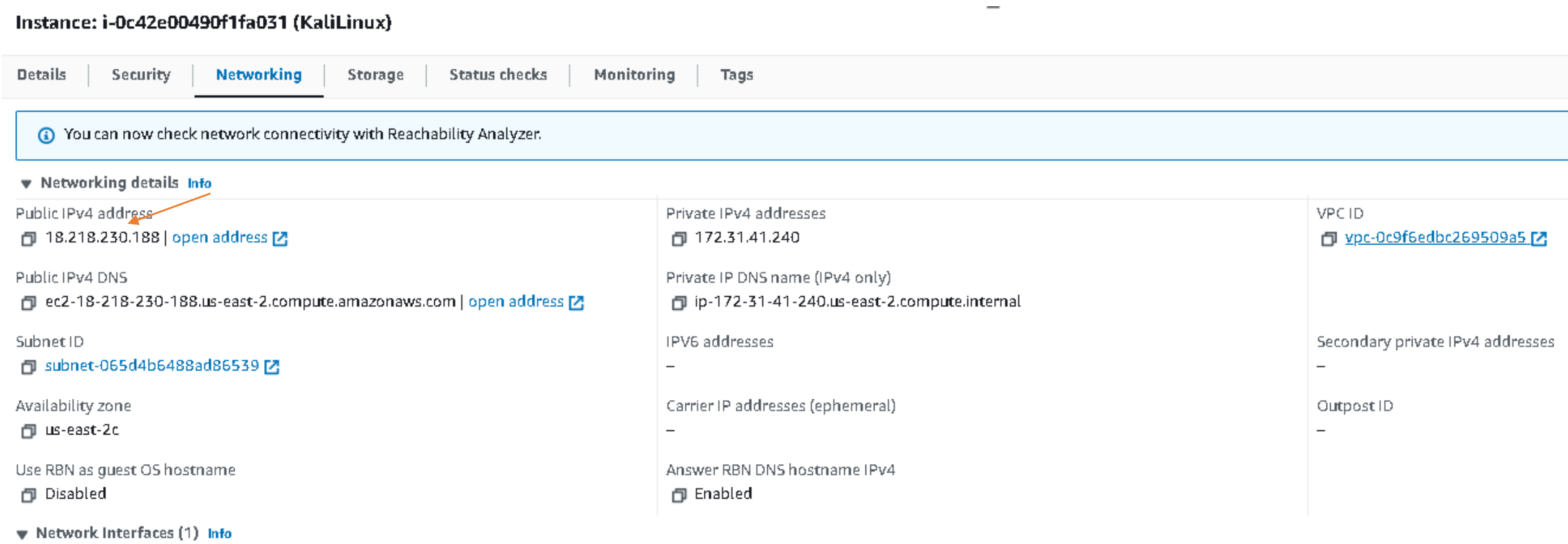


AWS Network Access Control List

- Previously,…
- In the security group, we try to limit the incoming IP address from public internet and we only allow IP addresses from AWS internal devices.
- We even try to limit (allow) the SSH connection by limiting the IP address only from “My IP Address”
- We even try to remove SSH inbound rule and we use AWS SSM agent to manage our EC2 instance

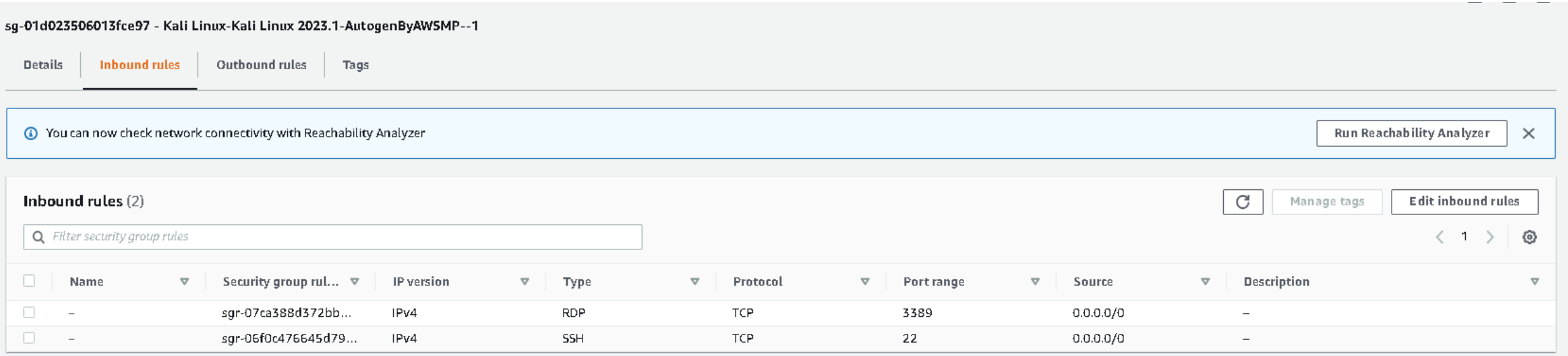
AWS Network Access Control List

- This is the public IP address for internet devices to communicate with our EC2 instance in the AWS cloud



AWS Network Access Control List

- Now we just leave this open “0.0.0.0/0”. Technically, all of the devices or machine in the internet can “visit” this instance.
- Because we have the setting allowing the incoming connection from 0.0.0.0/0



AWS Network Access Control List

- Let’s move over to the Kali Linux in my VirtualBox
- We use the “hydra” to attack our EC2 instance, dictionary attack, brute force.
- And there will be a fail in the attacks
- Check the next page

AWS Network Access Control List

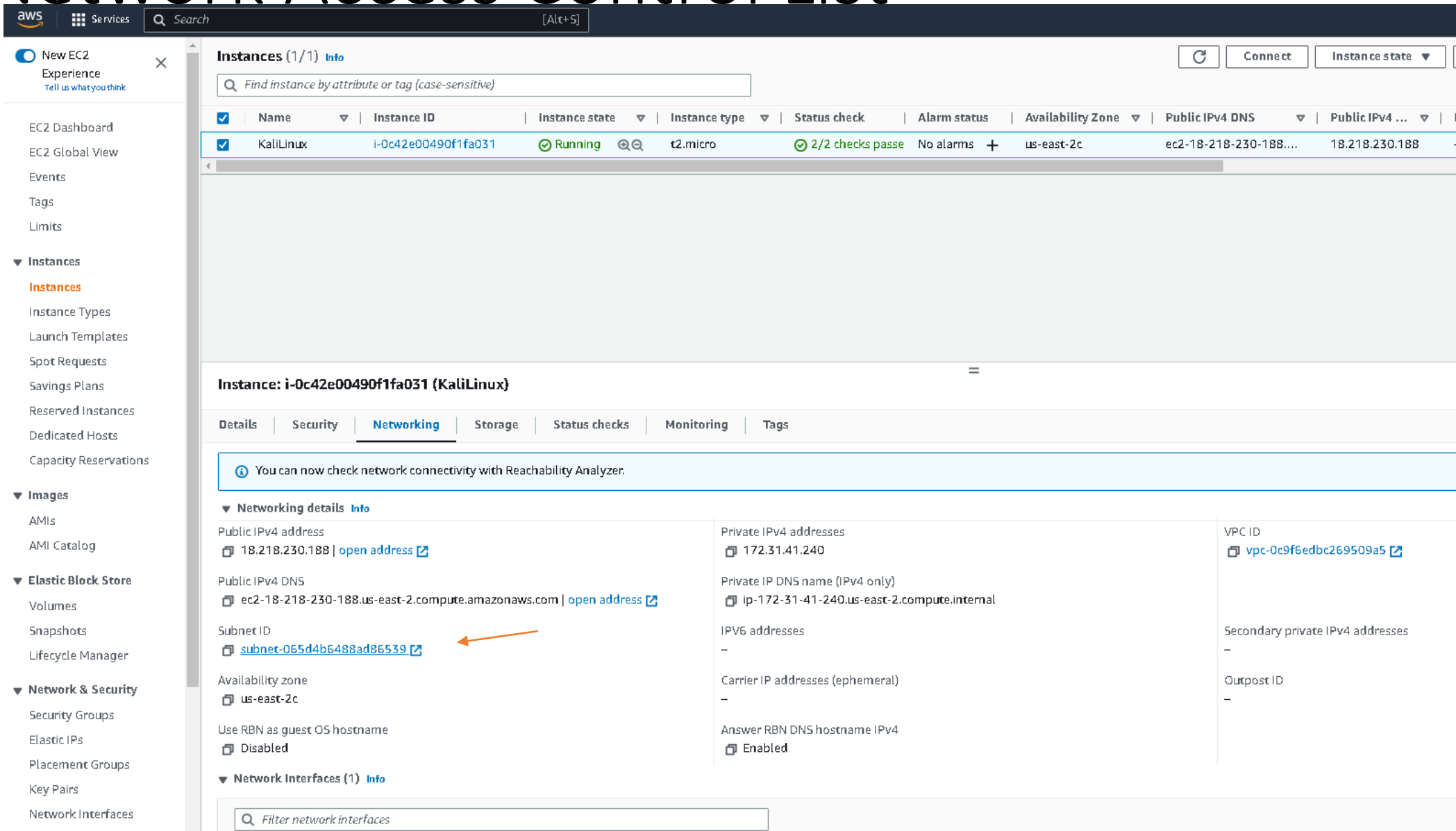
- This is very important for us! Why? Because we use key for protection.
- And you can also see I committed a silly mistake, the “rockyou.txt” is now in my home folder.
  - That’s why it is saying the file for password is not found

```
┌──(kali㉿kali)-[~]
└─$ hydra -l kali -P /usr/share/wordlists/rockyou.txt 18.218.230.188 ssh -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and eth
ics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-04-18 18:28:55
[ERROR] File for passwords not found: /usr/share/wordlists/rockyou.txt

┌──(kali㉿kali)-[~]
└─$ hydra -l kali -P ~/rockyou.txt 18.218.230.188 ssh -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and eth
ics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-04-18 18:30:28
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://18.218.230.188:22/
[ERROR] target ssh://18.218.230.188:22/ does not support password authentication (method reply 4).

┌──(kali㉿kali)-[~]
└─$
```

AWS Network Access Control List

- -t 4 for hydra command. This is saying I'm going to use 4 threads to perform brute force attack
- Using the private key file for authentication is correct.
- User name and password gives the hackers a good chance to hack into the system
  - Once they know your user name, there's just "one step" they can get into your account
- All right, so let's go back to the instances
- Check the next page for Subnet ID link. It is in "Networking" tab

AWS Network Access Control List



AWS Network Access Control List

- Here we go! The Network ACL is right there. (In the next page)
- Remember that in the NACL we can do setup to apply on the IP addresses you want to block further access from
- In the next page, we only have one rule (#100) which is to allow EVERYTHING.
- And there is a rule to **deny** all others
- Basically, it is still allowing EVERYTHING.

AWS Network Access Control List

aws | Services | Search [Alt+S] | Ohio | Charles Yu

VPC dashboard
EC2 Global View New
Filter by VPC:
Select a VPC

- Virtual private cloud
  - Your VPCs New
  - Subnets
  - Route tables
  - Internet gateways
  - Egress-only internet gateways
  - DHCP option sets
  - Elastic IPs
  - Managed prefix lists
  - Endpoints
  - Endpoint services
  - NAT gateways
  - Peering connections
- Security
  - Network ACLs
  - Security groups
- DNS firewall
  - Rule groups
  - Domain lists
- Network Firewall
  - Firewalls
  - Firewall policies
  - Network Firewall rule groups
  - Network Firewall

Subnets (1/1) Info

Actions | Create subnet

Filter subnets

Subnet ID: subnet-065d4b6488ad86539 | Clear filters

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses	Availability Zone	Av
–	subnet-065d4b6488ad86539	Available	vpc-0c9f6edbc269509a5	172.31.32.0/20	–	4090	us-east-2c	use

subnet-065d4b6488ad86539

Details | Flow logs | Route table | Network ACL | CIDR reservations | Sharing | Tags

Network ACL: acl-064142988a8b509cd

Edit network ACL association

Inbound rules (2)

Filter inbound rules

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Outbound rules (2)

Filter outbound rules

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

AWS Network Access Control List

- What we can do is to click this: Network ACL: acl-064142988a8b509cd
- So you can go into its detail setting for NACL
- Then, you can click its “Inbound rules”

AWS Network Access Control List

VPC dashboard
EC2 Global View New
Filter by VPC:
Select a VPC

- Virtual private cloud
  - Your VPCs New
  - Subnets
  - Route tables
  - Internet gateways
  - Egress-only internet gateways
  - DHCP option sets
  - Elastic IPs
  - Managed prefix lists
  - Endpoints
  - Endpoint services
  - NAT gateways
  - Peering connections
- Security
  - Network ACLs
  - Security groups
- DNS firewall
  - Rule groups
  - Domain lists
- Network Firewall
  - Firewalls
  - Firewall policies
  - Network Firewall rule groups
  - Network Firewall resource groups New

Network ACLs (1/1) Info

Actions | Create network ACL

Filter network ACLs

Network ACL ID: acl-064142988a8b509cd | Clear filters

Name	Network ACL ID	Associated with	Default	VPC ID	Inbound rules count	Outbound rules count	Owner
–	acl-064142988a8b5...	3 Subnets	Yes	vpc-0c9f6edbc269509a5	2 Inbound rules	2 Outbound rules	378253604690

acl-064142988a8b509cd

Details | Inbound rules | Outbound rules | Subnet associations | Tags

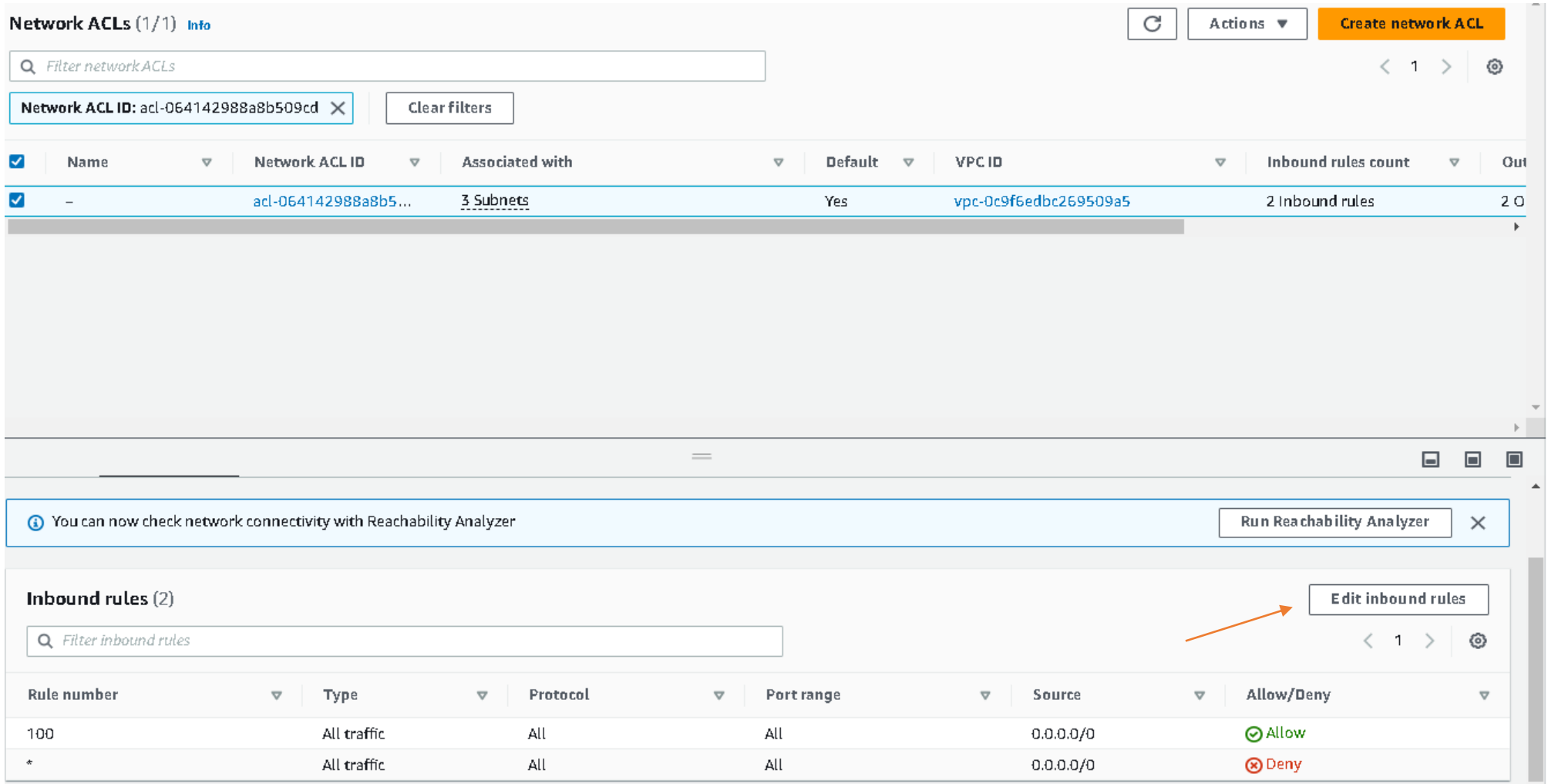
Details

Network ACL ID	Associated with	Default	VPC ID
acl-064142988a8b509cd	3 Subnets	Yes	vpc-0c9f6edbc269509a5

Owner
378253604690

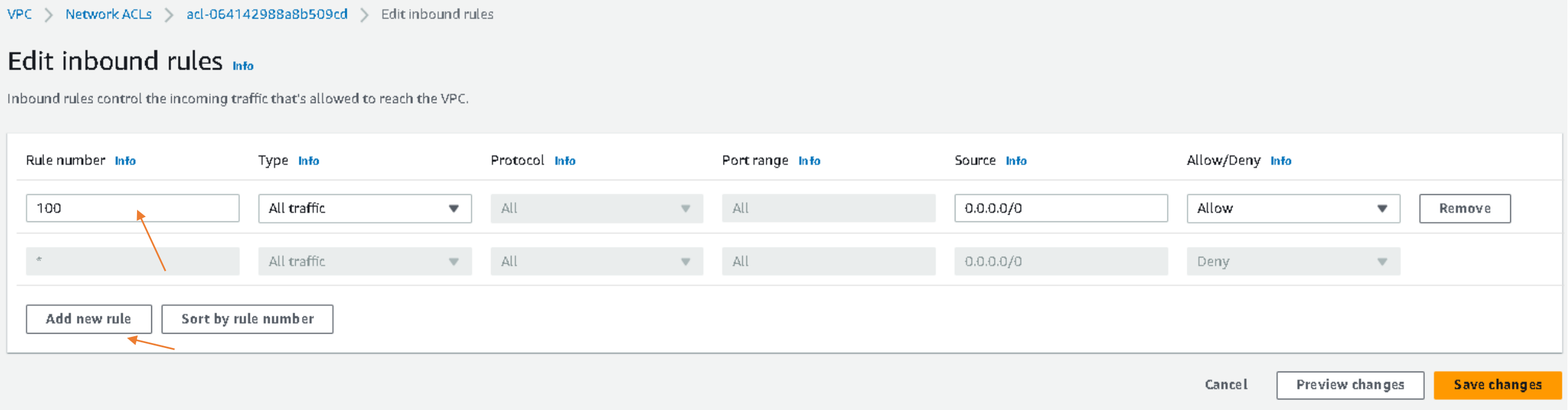
AWS Network Access Control List

- Now, click the [Edit inbound rules]



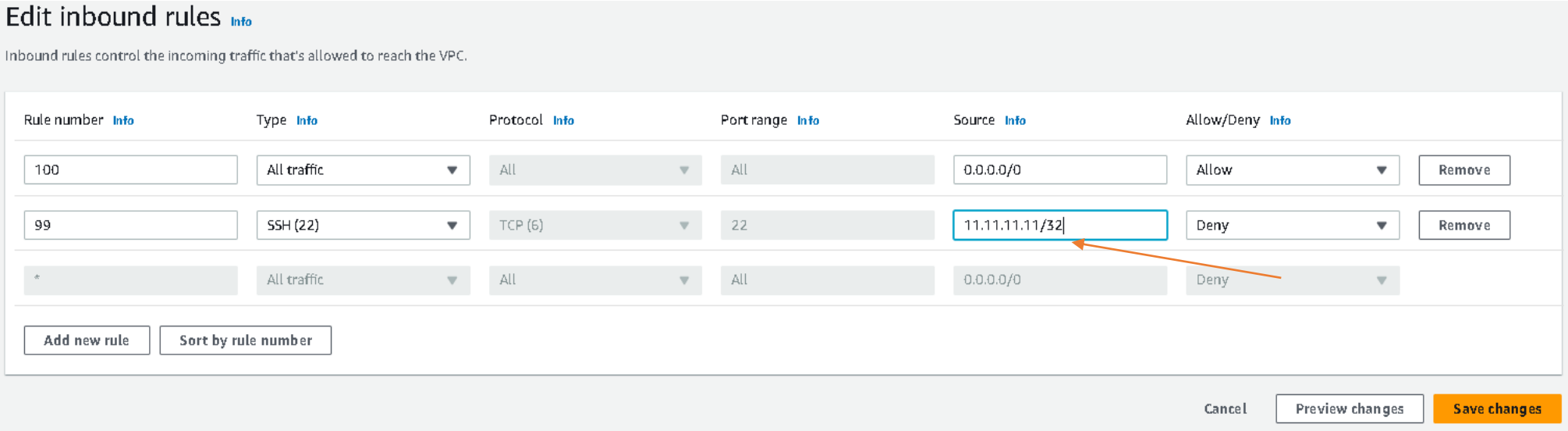
AWS Network Access Control List

- Once you click that, you have the following.
- What I'm going to do now is to "Add new rule"
- The 2nd row, "deny" is just "for your reference", which means, this is a template. It is existing there by default. (it is graying out)



AWS Network Access Control List

- Now what I'm going to do is to setup a "denial of connection" in the inbound rule
- Try to ask the Google "What is my IP address". ChatGPT doesn't know
  - It will tell you that your public IP address. i.e. My IP = 11.11.11.11
- Do the following settings and click the [Save changes]. Remember to put "/32"
- We use, SSH in our experiment



AWS Network Access Control List

- Now, you can see one more rule is produced

Inbound rules (3)

Edit inbound rules

Filter inbound rules

1

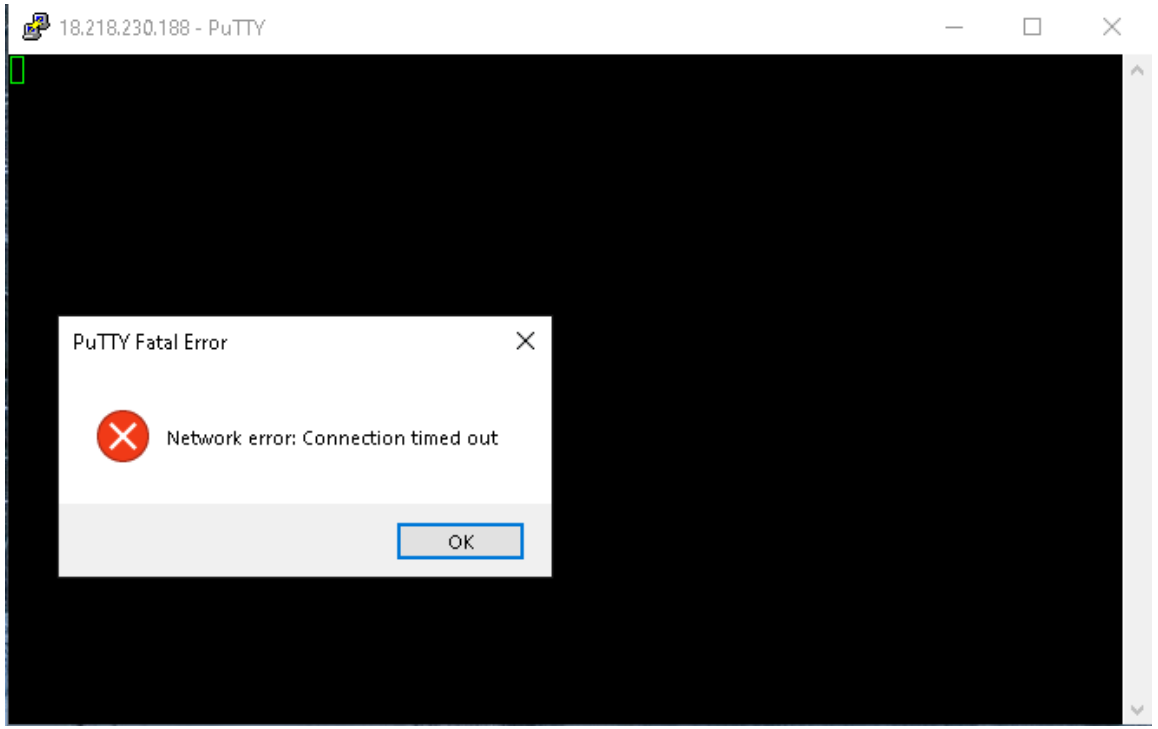
Rule number	Type	Protocol	Port range	Source	Allow/Deny
99	SSH (22)	TCP (6)	22	11.11.11.11/32	Deny
100	All traffic	All	All	0.0.0.0/32	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

- Rule #99 is just produced. It is denying a specific IP address for a specific service (SSH)
- Now if I jump back to the Putty (in my Windows) and try to connect again?

AWS Network Access Control List

- Here we go! We are blocked by ourselves! Even if I use my private key, I still cannot get in!

- Let’s try the “hydra” this time?



AWS Network Access Control List

- Did you see that? This time, the error message is slightly different.

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ hydra -l kali -P ~/rockyou.txt 18.218.230.188 ssh -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethi

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-04-18 21:37:59
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://18.218.230.188:22/
[ERROR] could not connect to ssh://18.218.230.188:22 - Timeout connecting to 18.218.230.188
```

- Could not connect to my EC2 instance in the AWS cloud via SSH
  - Because we are trying to brute force its SSH with dictionary file (rockyou.txt)
- Now the AWS EC2 instance is being protected by the newly created "deny rule" in the NACL against a specific IP address, which is our own IP address!