
AWS
re:Invent

SEC327

Instance containment techniques for effective incident response

Jonathon Poling

Principal Security Consultant

Threat Detection / Incident Response

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Instance compromise and containment scenario

The types (levels) of effective instance containment

An alternative to simple instance containment

Building your containment capabilities



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scenario: instance compromise



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scenario

Saturday
2 AM

Email: GuardDuty Alert - CryptoCurrency:EC2/BitcoinTool.B!DNS

“An EC2 instance is querying a domain name that is associated with cryptocurrency-related activity.”

Uh oh…

Scenario

Quick, to the Console!

Production Web Server? Uh oh…

Change the Security Group to “Isolation-SG”!

System contained!

Back to bed…

Scenario

Saturday
8 PM

Email: GuardDuty Alert - Trojan:EC2/DropPoint

“An EC2 instance is attempting to communicate with an IP address of a remote host that is known to hold credentials and other stolen data captured by malware.”

What??

Scenario

Sunday
8 AM

Compromised production server still running…
Network activity still occurring…

Shut down the instance as a last resort.

Start updating the résumé.

What went wrong here?

- ✗ We didn’t have a known working / tested plan for containment
- ✗ We didn’t understand security group rules and (expected) behavior
- ✗ We allowed for additional unnecessary dwell time within the environment by the attacker
- ✗ We very possibly shuttered a critical production server / service that will come with a substantial cost to the business
- ✗ We very possibly suffered sensitive data exfiltration

Why not just shut down / delete?

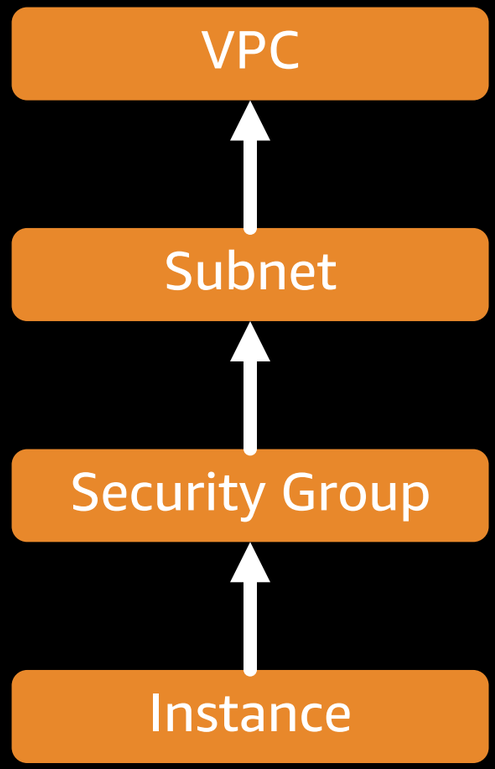
- Shutting down an instance achieves containment BUT comes at a cost
- You lose valuable (volatile) data from shutting down the instance
- This may significantly hinder your investigation and ability to perform a root cause analysis (RCA)
- Shut down / delete *may* make sense in some cases, *BUT*…
  - Shut down *should not* be a default or go-to response (in most cases)
  - It *should* be a purposeful and carefully considered / implemented decision

Then, what should I do?

- Understand the possibilities for containment
- Define containment procedures that are specific to the situation (threat type) *and* the resources (instances, data, etc.)
- Implement containment techniques that:
  - Prevent what you don’t want (and allow what you do)
  - Are effective at varying levels of granularity (from Instance to VPC)
  - Retain and derive as much useful data as possible for leverage in investigations and understanding scope

So, how do we do that?

We’ll work up the stack from the instance to the virtual private cloud (VPC)…



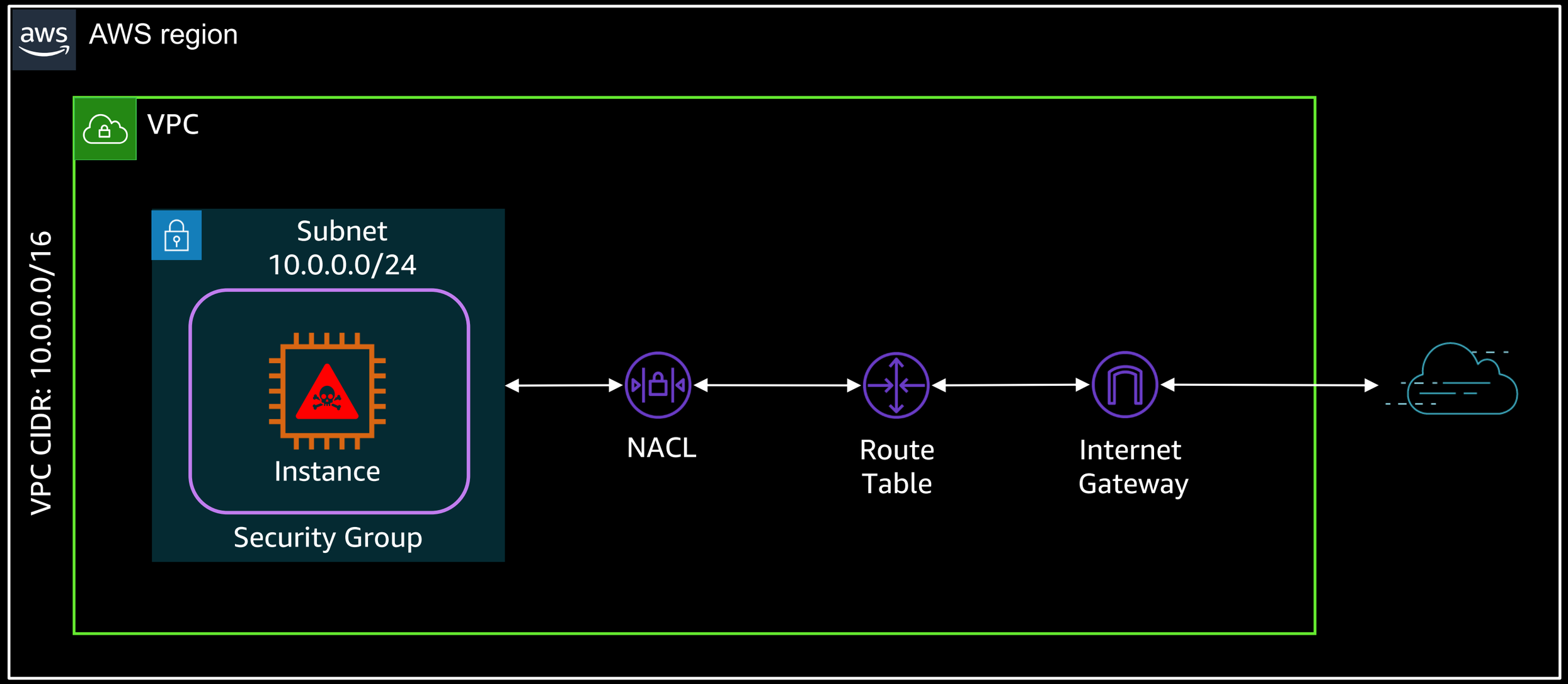
Reference architecture



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reference architecture



Instance containment techniques

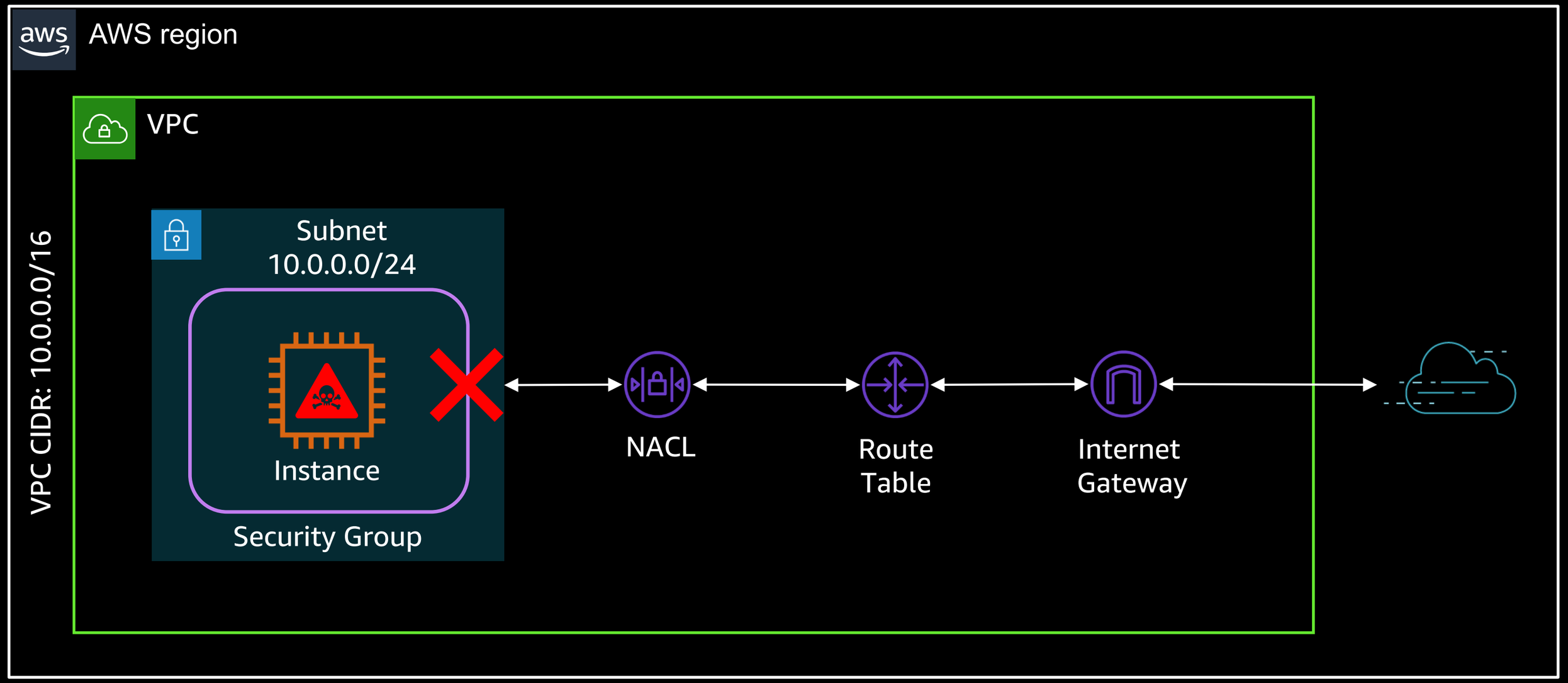


AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Security group–level containment

VIA SECURITY GROUP RULES



Security group–level containment

VIA SECURITY GROUP RULES

- Understanding security groups
  - Security group rules begin as an implicit Deny for all traffic (no rules)
    - You create rules to explicitly allow certain traffic
  - Responses to inbound traffic are allowed out regardless of outbound rules (and vice versa)
    - Via implementation of stateful connections
  - You can assign multiple security groups to an instance
    - If there is more than one rule for a specific port, *Amazon EC2 applies the most permissive rule*
  - You can add and remove rules at any time
    - Changes are immediately applied to the instances associated with the security group

      but…
  - The effect of some rule changes can depend on *how the traffic is tracked!*

Security group-level containment

VIA SECURITY GROUP RULES

! This is critical to implementing effective security group containment !

- Understanding “tracked” vs. “untracked” connections
  - Security groups use “connection tracking” to track information about traffic to / from the instance (i.e., implementing stateful connections)
  - But… *not all flows of traffic are tracked!* (Except ICMP traffic is always tracked)
- Untracked connections
  - Applies to traffic with a 0.0.0.0/0 rule AND a 0.0.0.0/0 (0-65535) rule in the other direction
    - Inbound 0.0.0.0/0 and Outbound 0.0.0.0/0 (0-65535) – and vice versa
  - Flow of traffic is *immediately interrupted* if the rule that enables the flow is removed or modified (if Inbound rule for Inbound connection is removed/modified, and vice versa)
- Tracked connections
  - Applies to traffic with specific IP / CIDR rules (e.g., 203.0.113.1/32, 203.0.113.0/24, …)
  - Flow of traffic *is not interrupted* if the rule that enables the flow is removed or modified, regardless of rules in the opposite direction

Security group-level containment

VIA SECURITY GROUP RULES

T = Tracked connection

U = Untracked connection

Inbound rules

	Protocol type	Port number	Source IP
T	TCP	22 (SSH)	203.0.113.1/32
U	TCP	80 (HTTP)	0.0.0.0/0
U	TCP	80 (HTTP)	::/0
T	ICMP	All	0.0.0.0/0

Outbound rules

	Protocol type	Port number	Destination IP
U	All	All	0.0.0.0/0
U	All	All	::/0

Removing this rule **WILL NOT** terminate existing connections, but **WILL** prevent any future inbound connections

Removing either rule **WILL** immediately terminate all existing inbound HTTP connections and prevent any future inbound connections

ICMP is always tracked, regardless of the presence of any rules

Removing either rule will make all new respective Inbound traffic Tracked

Security group–level containment

VIA SECURITY GROUP RULES

Pros

- ✓ Most granular type of containment
- ✓ Provides effective and targeted containment of a single instance*

*When using separate / dedicated security groups for isolation

Cons

- X Rules can terminate only “untracked” connections, not “tracked” connections
- X Can’t just perform a one-click application of a prebuilt ”Isolation” security group with no rules to prevent traffic
- X Requires a multistep process to both terminate active connections and prevent future connections
- X If not done exactly as described, traffic will still be (unknowingly) allowed

Security group–level containment

VIA SECURITY GROUP RULES

- How-to (via existing attached security group)
  - Identify the security group associated with the instance
  - Delete all existing rules
  - Create a single rule of 0.0.0.0/0 (0-65535) for all traffic in *both* the inbound rules and outbound rules
    - This converts all existing (and new) traffic to being ”untracked"
  - Remove the 0.0.0.0/0 (0-65535) inbound and outbound rules to terminate all (now all ”untracked”) connections

Security group-level containment

VIA SECURITY GROUP RULES

- How-to (via separate / dedicated security group)
  - Option 1: One separate / dedicated security group
    - Create a dedicated “Isolation” security group
    - Create a single rule of 0.0.0.0/0 (0-65535) for all traffic in *both* the inbound rules and outbound rules
      - Application of these rules will convert all existing (and new) traffic to ”untracked”
    - Remove the existing security group association from the instance and associate the “Isolation” security group with the instance
    - Delete both rules of 0.0.0.0/0 (0-65535) for all traffic from both the inbound rules and outbound rules of the “Isolation” security group

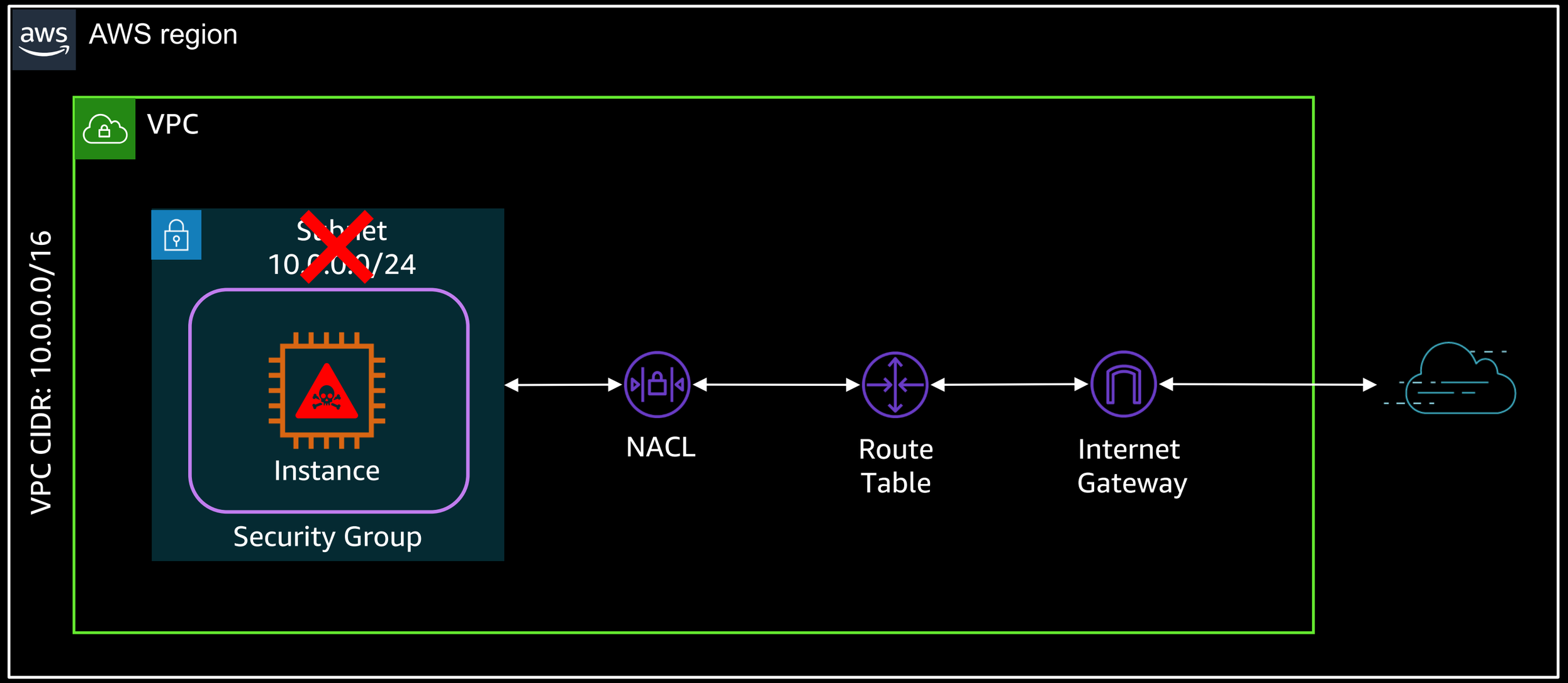
Security group-level containment

VIA SECURITY GROUP RULES

- How-to (via separate / dedicated security group)
  - Option 2: Two separate / dedicated security groups
    - Create a dedicated “IsolationSG-Step1”security group
      - Create a single rule of 0.0.0.0/0 (0-65535) for all traffic in *both* the inbound rules and outbound rules
        - Application of these rules will convert all existing (and new) traffic to ”untracked”
    - Create a dedicated “IsolationSG-Step2” security group with no rules
    - Remove the existing security group association from the instance and associate the “IsolationSG-Step1” security group with the instance
      - This converts all existing (and new) traffic to ”untracked"
    - Remove the “IsolationSG-Step1” security group association from the instance and associate the “IsolationSG-Step2” security group with the instance

Subnet-level containment

VIA “ISOLATION” SUBNET



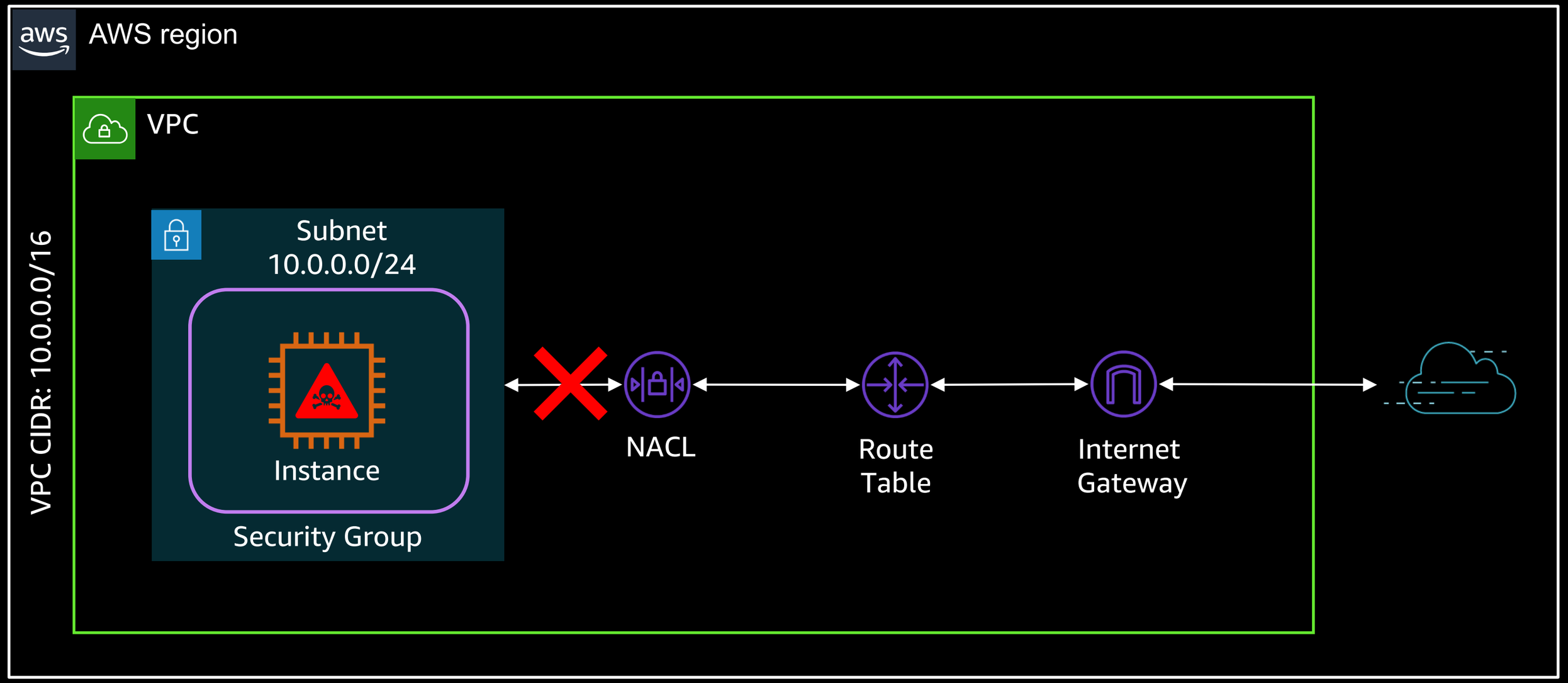
Subnet-level containment

VIA “ISOLATION” SUBNET

- Can we just create an “Isolation” subnet and move compromised instances into it?
  - Unfortunately, no
- Can’t change the subnet associated with a running instance
  - Must shut down the instance before changing the subnet association
- But, is all hope lost?
  - Nope, we’ll investigate how to leverage an “Isolation” subnet later for monitoring

Subnet-level containment

VIA NETWORK ACLS (NACLS)



Subnet-level containment

VIA NETWORK ACLS (NACLS)

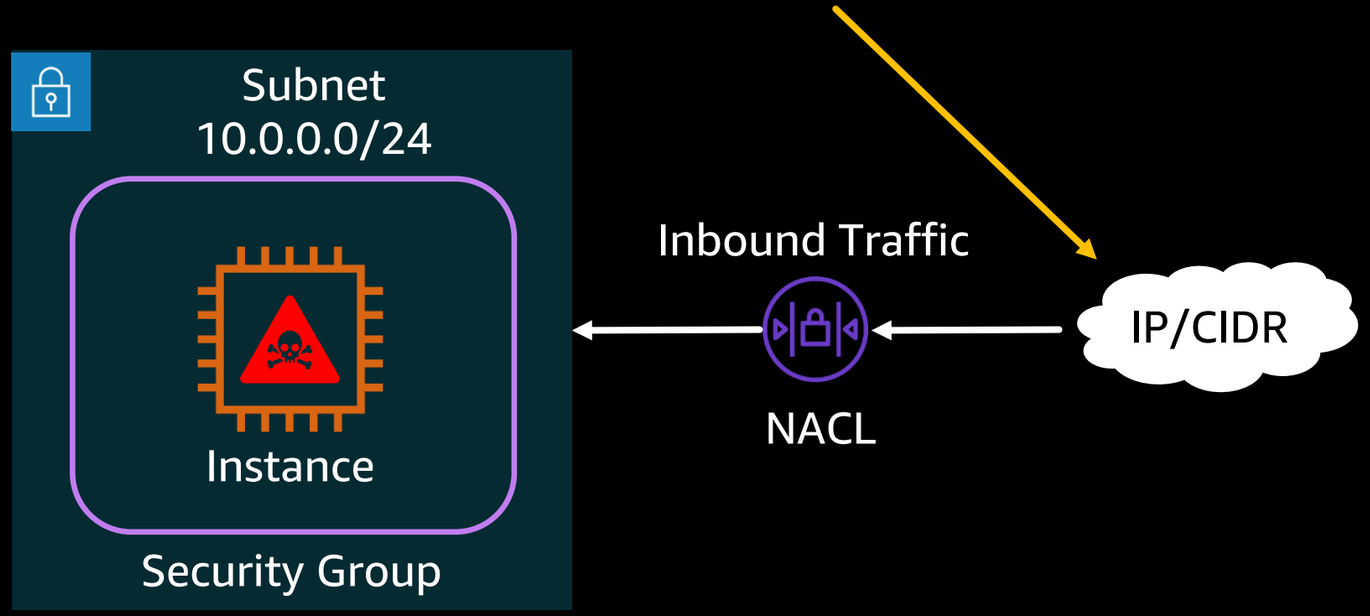
- Understanding Network ACLs (NACLs)
  - Denies / allows traffic solely based on external (to the Subnet) IP/CIDR
    - For inbound rules, must specify “source” (IP/CIDR external to Subnet)
    - For outbound rules, must specify “destination” (IP/CIDR external to Subnet)
  - Cannot deny / allow traffic based on internal IP/CIDR (e.g., instance IP)
  - Stateless (doesn’t matter if it’s a response to allowed traffic)
  - Each NACL (and included rules) can be associated with only one subnet at a time
  - Maximum 20 rules per NACL (20 IPv4 rules plus 20 IPv6 rules)
  - Maximum 200 NACLs per VPC

Subnet-level containment

VIA NETWORK ACLS (NACLS)

Inbound Rule

Allow/Deny traffic based on “source”

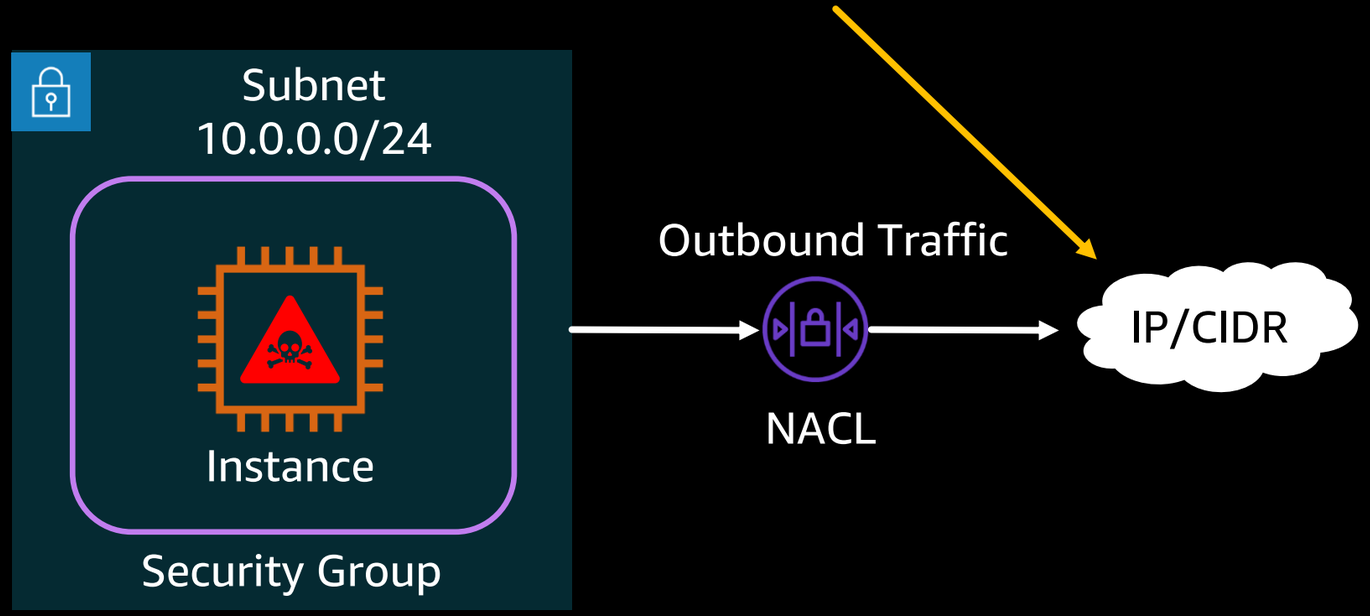


Subnet-level containment

VIA NETWORK ACLS (NACLS)

Outbound Rule

Allow/Deny traffic based on “destination”



Subnet-level containment

VIA NETWORK ACLS (NACLS)

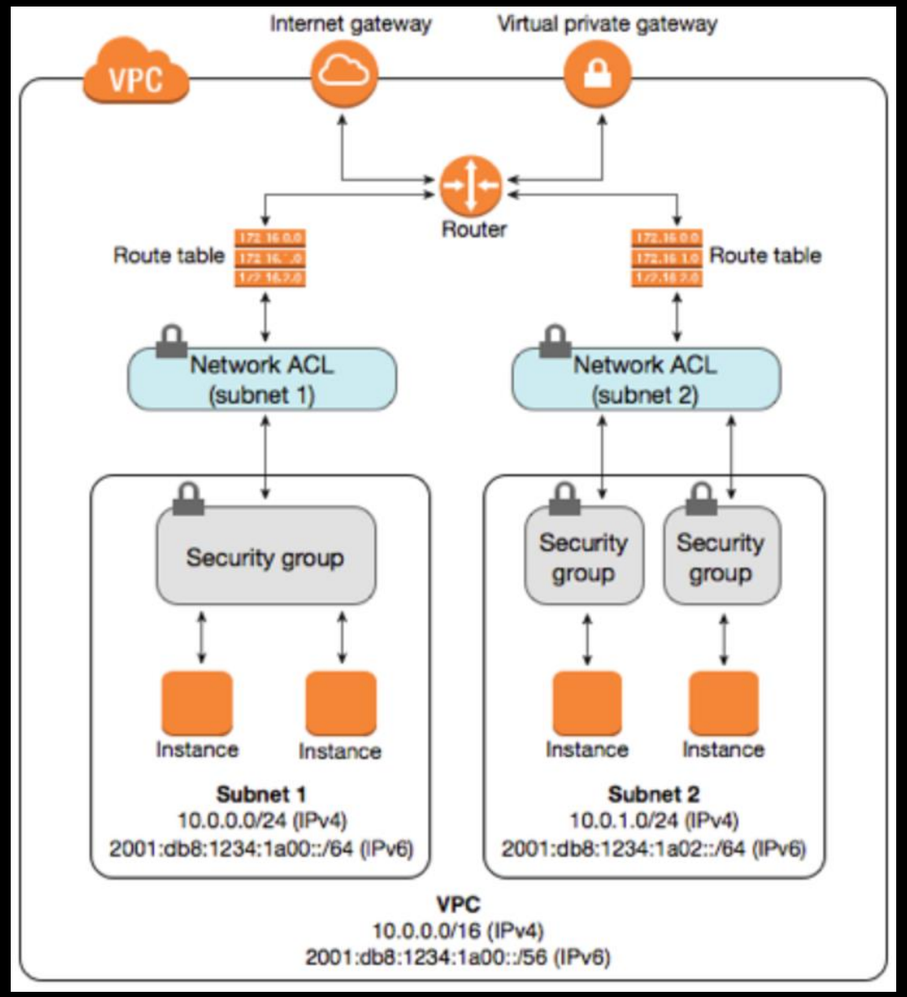
Security Groups vs. NACLs

Security group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in order, starting with the lowest numbered rule, when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets that it's associated with (therefore, it provides an additional layer of defense if the security group rules are too permissive)

Subnet-level containment

VIA NETWORK ACLS (NACLS)

Security Groups vs. NACLs



Subnet-level containment

VIA NETWORK ACLS (NACLS)

Pros

- ✓ Takes just a single inbound and outbound NACL rule to both terminate existing connections and prevent future connections

Cons

- X Cannot be used in a targeted fashion to isolate a single instance
- X Can block only specific external IPs / CIDRs
- X Isolates all instances on the associated subnet*

  *Could be a “pro” if you’re looking to isolate an entire subnet
- X May require deleting an existing NACL rule in order to fit the Isolation NACL rule (if NACL rules are at maximum capacity)

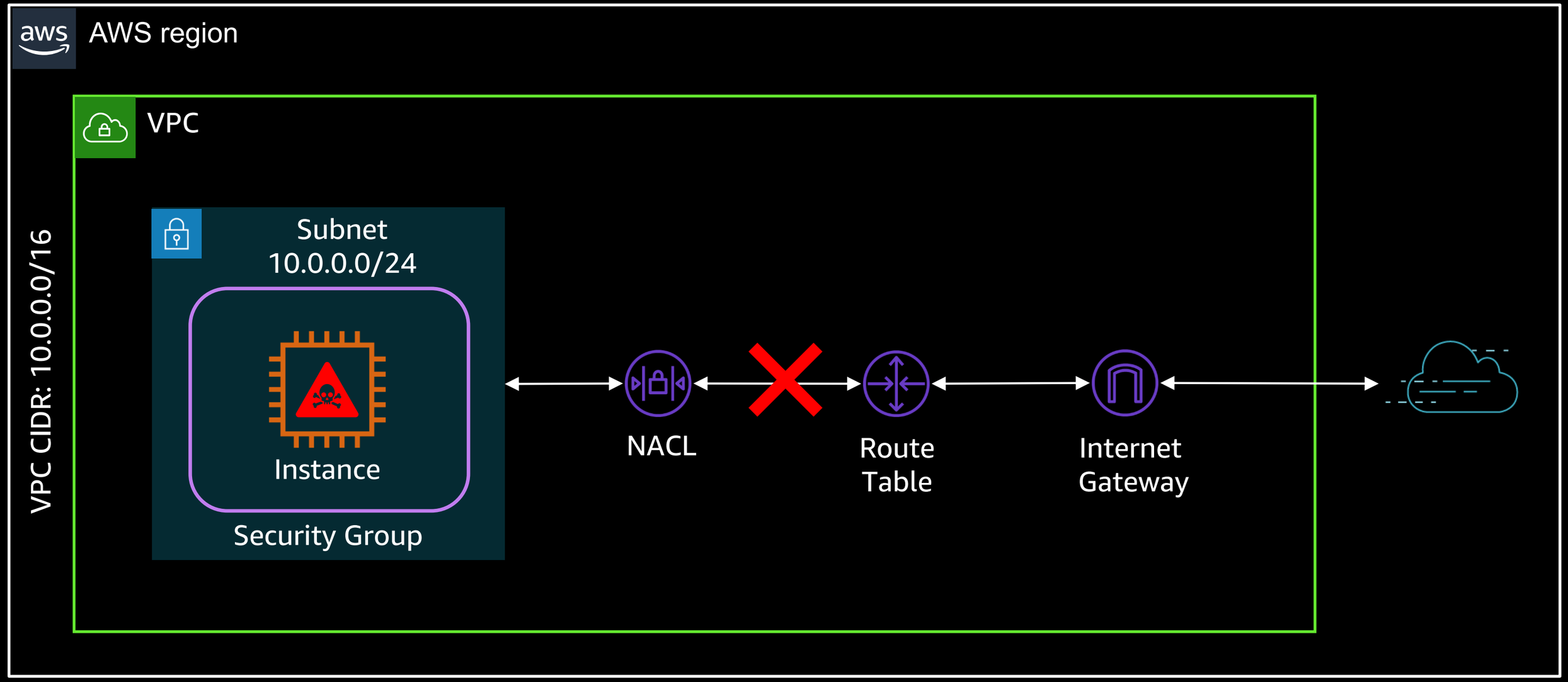
Subnet-level containment

VIA NETWORK ACLS (NACLS)

- How-to (via existing NACL)
  - Identify the subnet associated with the instance
  - Identify the NACL associated with the subnet
  - Add a DENY ALL NACL rule to both the inbound and outbound rules as rule number 1 for all traffic (0.0.0.0/0)
    - If you need to delete an existing rule to make space, ensure you record the details of the deleted rule in order to restore it in the future
- How-to (via new NACL)
  - Identify the VPC and subnet associated with the instance
  - Create a new NACL within the VPC where the instance resides
    - By default, the NACL will create a single DENY ALL rule for 0.0.0.0/0 within both the inbound rules and outbound rules
  - Associate the subnet of the instance with the new NACL

Subnet-level containment

VIA ROUTE TABLE



Subnet-level containment

VIA ROUTE TABLE

- Understanding route tables
  - Route tables define and control the routing for all of your VPC subnets
  - Types of route tables
    - Main route table
      - Created automatically when you create a VPC
      - Controls the routing for all subnets that are not explicitly associated with any other route table
    - Custom route tables
      - Non-default route tables you can customize (created empty with no routes by default)
  - Each subnet in your VPC must be associated with a route table
    - Cannot associate a subnet with more than one route table (1:1)
    - Subnets that aren't explicitly associated with any route table have an implicit association with the main route table
  - Can optionally associate a route table with an internet gateway (IGW) or a virtual private gateway (VGW)

Subnet-level containment

VIA ROUTE TABLE

Pros

- ✓ Minimal intervention to perform containment
- ✓ Simple application (association) of a route table to a subnet
- ✓ Can have a prebuilt “Isolation” route table that’s ready to go

Cons

- X Cannot be used in a targeted fashion to isolate a single instance
- X Isolates all instances on the associated subnet*

  *Could be a “pro” if you’re looking to isolate an entire subnet
- X Still allows intra-VPC (local) routing (i.e., Instance-to-Instance traffic)

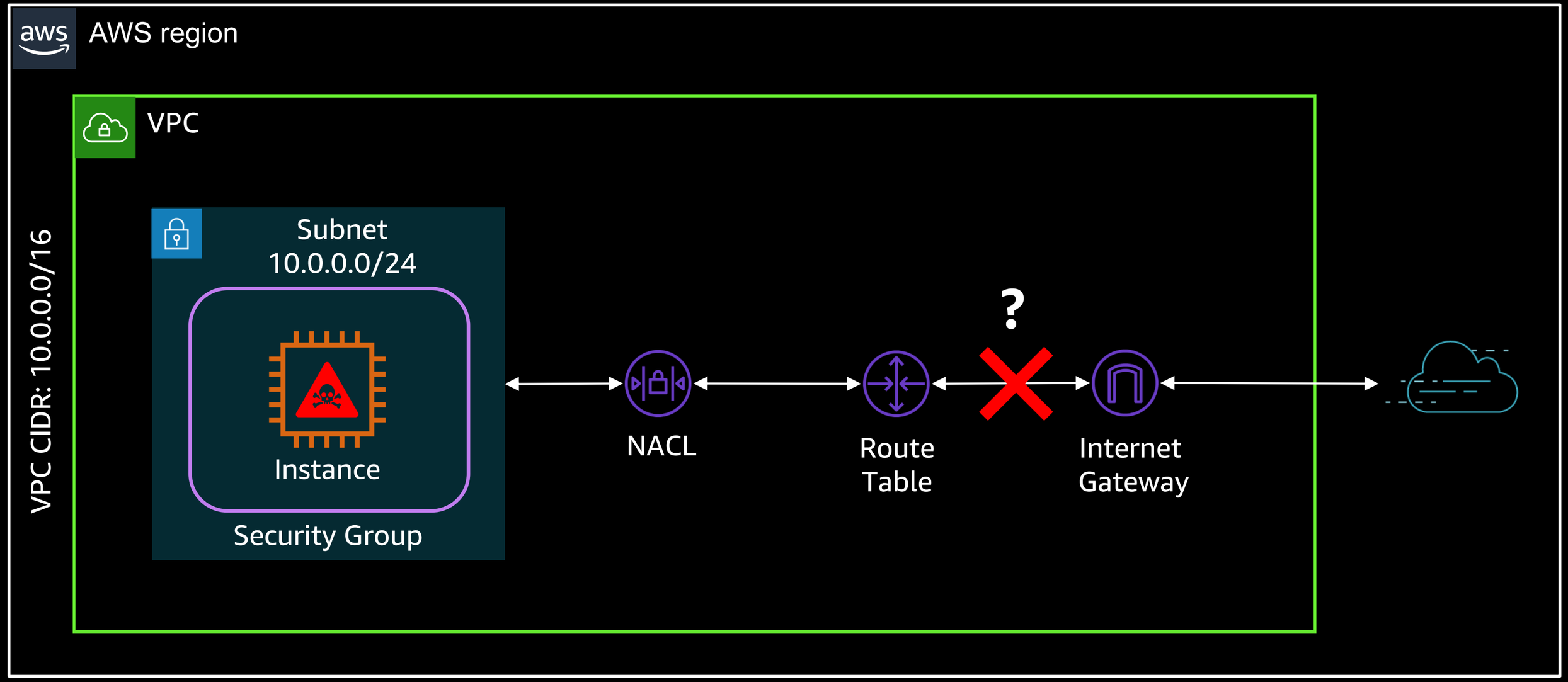
Subnet-level containment

VIA ROUTE TABLE

- How-to
  - Create a custom route table (will be empty with no routes by default)
  - Identify the subnet associated with the instance
  - Associate the custom route table with the subnet of the instance

VPC-level containment

VIA INTERNET GATEWAY (IGW)



VPC-level containment

VIA INTERNET GATEWAY (IGW)

- Understanding internet gateways (IGWs)
  - Serves two main purposes
    - Provide a target in your VPC route tables for internet-routable traffic
    - Perform network address translation (NAT) for instances that have been assigned public IPv4 addresses
  - Supports both IPv4 and IPv6 traffic

VPC-level containment

VIA INTERNET GATEWAY (IGW)

- Can we just detach the IGW from the VPC to isolate everything?
  - Unfortunately, no
- You will get dependency errors if you attempt to do this in an active environment
  - Amazon EC2 instances with public IP addresses / mappings will prevent you from detaching the IGW until those mappings are removed (i.e., instances are shut down)
  - Shutting down every available instance in a VPC for isolation is clearly not viable (nor wanted)
- So, how would we do VPC-level containment?
  - Remove all the IGW routes from all route tables
  - Attach a custom route table with no routes to all subnets within the VPC
  - Options available depending on your needs and the situation
  - This is a *big* hammer: Use it wisely

But… what if…

You want to isolate the instance *but* also want to

- Maintain direct access (SSH/RDP) to the instance
- Allow for continued AV / EDR / telemetry monitoring and reporting
- Acquire volatile data (memory)
- Collect additional indicators of compromise (IOCs) from the instance



Instrumented monitoring via isolation VPC



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Isolation VPC

INSTRUMENTED MONITORING

- What is an ”Isolation” VPC?
  - An “Isolation” VPC is a VPC you preconfigure* to serve as an environment to facilitate:
    - Real-time system access
    - Real-time system and network monitoring
    - Collection of information / intelligence from the compromised instance(s)
  - Ideally located within a dedicated and effectively instrumented / limited Forensics account

Isolation VPC

INSTRUMENTED MONITORING

- Why utilize one?
  - Isolating an instance involves terminating all active connections
  - This means possible loss of valuable information such as:
    - Source IP(s) of actively connected attacker(s)
    - IPs / domains involved in command and control (C&C) activity
    - Network data / payloads being actively transferred
    - Commands / activities being performed on the system
  - These indicators of compromise (IOCs) can be extremely valuable in helping to:
    - Identify additionally compromised systems
    - Build comprehensive detection / prevention measures for the future

Isolation VPC

INSTRUMENTED MONITORING

Pros

- ✓ Provides effective containment while also allowing for *carefully* instrumented access
- ✓ Continued leveraging of AV / EDR as a force multiplier for containment / response
- ✓ Live / interactive querying and collection of data from the host
- ✓ Collection of additional IOCs for use in proactively identifying additional possibly compromised instances / infrastructure

Cons

- X Must know the full set of IPs and ports utilized by your endpoint utilities you wish to continue communicating
- X Must know the specific sources (IP, subnet, etc.) of connections from which you plan to connect to the host for interactive querying
- X Must be very comfortable building and implementing security groups, NACLs, and routes to ensure the compromised instance's communications are properly limited

Isolation VPC

INSTRUMENTED MONITORING

Considerations when leveraging an Isolation VPC:

- The instance will have to be shut down before it can be moved to the Isolation VPC
- Source IP(s) of actively connected attackers will need to be collected prior to instance shutdown as those will not be restarted / reinstated (for good reason)
- The efficacy of monitoring will rely on the malware / activity persisting across reboots

Do not continue running a compromised instance if there is a possibility that it may propagate infection or facilitate additional unauthorized data access

Isolation VPC

INSTRUMENTED MONITORING

- What's the process for instrumented isolation and monitoring?
  - Build isolation network (example)
    - Create a VPC (ideally within a dedicated Forensics account)
    - Create two subnets
      - "Isolation" subnet
      - "Analysis" subnet
    - Create an "Isolation" security group that allows:
      - Outbound connections to monitor malware C&C (port 53, 80, 443, etc. traffic – TBD by you)
      - Outbound connections to AV / EDR endpoints for continued communications / detection updates
      - Outbound connections for VXLAN traffic (UDP port 4789) to "analysis" security group
      - Inbound SSH/RDP from analyst subnet for live querying / monitoring
    - Create an "Analysis" security group that allows:
      - Outbound connections to "Isolation" security group to connect to compromised instance(s)
      - Inbound connections for VXLAN traffic (UDP port 4789) from "Isolation" security group for monitoring

Isolation VPC

INSTRUMENTED MONITORING

- What’s the process for instrumented isolation and monitoring?
  - Instrument network data collection (one or both – your choice)
    - Traffic mirroring (for full PCAP collection)
      - Setup a Nitro-based “Monitoring” instance in the “Analysis” subnet as a traffic mirror target (receiver)
      - Instrument the instance with appropriate tooling to receive / capture / log the mirrored traffic
    - VPC flow logs (for flow record collection)
      - Enable VPC flow logs for the Isolation VPC
  - Instrument analysis instance(s)
    - Launch and preconfigure the appropriate instances within the “Analysis” subnet for querying the isolated instance and performing forensic analysis

Isolation VPC

INSTRUMENTED MONITORING

- What's the process for instrumented isolation and monitoring?
  - Execute the isolation and monitoring
    - For the compromised instance
      - Collect any necessary volatile data
      - Shut down the instance
      - Create an image (AMI) from the instance
        - (Optional) Copy the AMI to another region, if needed
      - Share the AMI (via Modify Image Permissions) with the Forensics account
    - Within the Forensics account
      - Launch the AMI as a Nitro-based instance within the "Isolation" subnet of the Isolation VPC
      - Create a traffic mirror session with the following:
        - Traffic mirror target: ENI of the "Monitoring" instance in "Analysis" subnet
        - Traffic mirror source: ENI of the compromised instance in "Isolation" subnet
        - Traffic mirror filter: Inbound and outbound rules for *all* TCP and UDP traffic for 0.0.0.0/0

Isolation VPC

INSTRUMENTED MONITORING

- Leverage the benefits of containment PLUS real-time data sources!
  - From full packet capture:
    - ✓ Identify C&C activity and commands being issued to understand attack(er) scope
    - ✓ Derive and/or extract encryption and decryption keys to decrypt communications/data
    - ✓ Extract files and binaries (executables) for network IDS signatures
  - From system access:
    - ✓ Identify malicious processes (names and hashes) for supplemental AV / EDR signatures
    - ✓ Identify attacker commands run on the system for insights into attacker's intent / goals
    - ✓ Identify malicious files and binaries created for hashing and org-wide search / detection
    - ✓ Collect and reverse-engineer the malware to derive additional IOCs for detection

Building your containment capabilities



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building your containment capabilities

STEPS FOR BUILDING EFFECTIVE CONTAINMENT TECHNIQUES

Crawl

1. Establish a dedicated Forensics account and test (sandbox) environment
2. Understand the levels of containment (Security Group vs. Subnet vs. VPC)
3. Define containment procedures based on Alert, Resource, Environment, Data, etc.
4. Define failover plans for if / when a containment technique isn’t effective

Walk

5. Build and test the basic containment techniques in a sandbox environment
6. Implement the basic containment techniques within production

Run

7. Build and test the Isolation VPC in a sandbox environment
8. Implement the Isolation VPC within your Forensics Account
9. Test your plans (and expectations) regularly

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.