CSCI 5902 Advanced Cloud Architecting (Fall 2023)
Assignment 4 – Networking

Name : Yogish Honnadevipura Gopalakrishna
Banner ID : B00928029

1. To handle outbound internet connectivity from the private subnets, I propose deploying a NAT Gateway in every subnet in all VPCs globally.

Shay's decision to deploy a NAT Gateway in every subnet globally for handling outbound internet connectivity has both benefits and drawbacks. The primary advantage of this approach lies in ensuring that instances in private subnets across the globe can independently access the internet, reducing the risk of a single point of failure and potentially improving scalability during usage spikes. However, this decision comes with notable drawbacks.

One significant concern is the potential increase in costs associated with deploying NAT Gateways in every subnet. Given that NAT Gateways are billed based on usage, having multiple instances globally may lead to higher expenses for Cupid. Moreover, the complexity of managing NAT Gateways in each subnet could pose challenges in terms of configuration, monitoring, and troubleshooting. Coordinating changes across numerous instances distributed globally might introduce operational difficulties.

A suggested alternative approach is to consider a centralized NAT Gateway strategy[1]. By deploying a single or a small number of NAT Gateways in a central location, such as a dedicated subnet or VPC, Cupid could potentially mitigate some of the drawbacks associated with Shay's proposal. This centralized approach can offer cost savings, streamline management tasks, and facilitate more efficient monitoring and maintenance.

The difficulties associated with outbound internet access can be addressed by Cupid by using a centralised NAT Gateway solution and carefully examining the network architecture. This will optimise expenses, streamline management, and making sure it's scalable.

2. Given that we have multiple AWS regions in use, I suggest creating a fully meshed VPC peering setup for all VPCs in each region.

It is not advised to establish a fully meshed VPC peering setup for every VPC across several AWS regions because of the possible complexity, scalability issues, and performance consequences that could arise. As the number of VPCs and regions increases, there is an increasing number of peering connections to handle, which adds to the complexity. Every VPC needs to create peering connections with every other VPC, which results in a large administrative burden, possible setup errors, and troubleshooting challenges.

The scalability of this strategy is hampered by the exponential growth of peering connections, which makes it difficult to maintain and scale the network architecture efficiently. It becomes difficult to add new VPCs or regions, adding significant complexity to configuration and management. Moreover, the performance in a fully meshed VPC peering

setup may suffer, with suboptimal traffic paths causing increased latency and potential bottlenecks, leading to delays in data transfer.
The implementation of a hub-and-spoke network structure is a more advised strategy. To do this, a central "hub" VPC must be created in each area, and VPC peering connections must be established between the hub and "spoke" VPCs[3]. This architectural model reduces the amount of peering connections that require configuration and upkeep, which simplifies management. Scalability benefits of the hub-and-spoke architecture include the simple addition of new VPCs or regions through peering links with the relevant hub VPC. Because the hub VPC serves as a centralised location for putting network access control and security measures into place, it offers improved control and security.

Due to traffic passing through the hub VPC, the hub-and-spoke architecture adds some extra latency; however, this can be reduced by choosing instances for the hub VPCs that are suitably sized and networked. For Cupid's scattered infrastructure spanning several AWS regions, the hub-and-spoke network topology provides a more effective, scalable, and manageable solution overall.

3. To ensure the security of our sensitive user data, I propose encrypting all traffic in-transit within our VPCs using IPsec tunnels

Using IPsec tunnels to encrypt all traffic in-transit within VPCs may not be the best course of action, given the strong security safeguards built into AWS's VPC infrastructure. AWS provides a full range of security capabilities that are specifically made to improve data protection in VPCs. These safeguards include private communication using AWS Private Linecure VPN connections, and TLS termination in CloudFront and ELB services.

Cupid may effectively secure data in-transit without the extra complexity and overhead involved with establishing IPsec tunnels for every VPC traffic by utilising AWS's existing encryption features[2]. The current encryption systems in place inside AWS services are made to adhere to strict security guidelines and provide strong security for private user information.

Implementing IPsec tunnels to encrypt all traffic inside VPCs could add needless complexity and possibly have an adverse effect on performance. AWS's well-established encryption technologies are well-suited to guarantee data security and privacy without sacrificing performance because they are integrated into their services and features. Encrypting user data while it's in transit and creating a secure environment within VPCs are two benefits of using AWS's encryption features that comply with best practises.

Using SSL/TLS encryption for sensitive data transferred over the internet, utilising security groups and Network Access Control Lists (NACLs) for access control, and employing VPN connections for secure connectivity between on-premises infrastructure and VPCs are all part of concentrating on AWS's encryption capabilities. Cupid may create a secure virtual private cloud (VPC) environment that complies with data privacy standards and guarantees the confidentiality and integrity of sensitive customer data while in transit by leveraging AWS's sophisticated security measures and adhering to their best practises.

4. I propose using the same NACLs for all our subnets to maintain a consistent security posture.

A recommended approach would be to carefully evaluate the security requirements of each subnet within Cupid's Virtual Private Cloud (VPC) and create specific Network Access Control Lists (NACLs) tailored to their unique needs. While the proposal to use the same NACLs for all subnets provides consistency, a more nuanced approach acknowledges that subnets may host diverse services with distinct security considerations.

By assessing the specific security requirements of each subnet, Cupid can design NACLs that align with the characteristics of the services hosted within them. This tailored approach allows for flexibility, accommodating different levels of access control based on the sensitivity of the data or the nature of the applications. For example, subnets handling sensitive user data may require more restrictive rules compared to subnets hosting less critical services.

Creating subnet-specific NACLs provides the granularity needed to address the unique security postures of each subnet. This approach ensures that security measures are aligned with the specific demands of different parts of the infrastructure. It allows Cupid to strike a balance between maintaining a standardized security framework and adapting to the varied security requirements across its VPC.

While this approach may introduce additional configuration overhead compared to using identical NACLs for all subnets, the trade-off is a more robust and tailored security posture. It enables Cupid to establish a nuanced and effective security architecture that considers the specific needs of its diverse range of services. In conclusion, the recommended approach involves a careful evaluation of security requirements for each subnet and the creation of specific NACLs to meet those needs, providing a balanced and adaptive security framework for Cupid's VPC.

5. To enable secure access to user profile from the VPCs, I propose using public APIs which is very convenient.

Using public APIs to enable secure access to user profiles from the VPCs may not be the best course of action given the security ramifications and data privacy concerns. Public APIs usually face the public internet and are designed primarily for external access. Relying on public APIs for internal access to sensitive user profile data within the VPCs poses security issues, potentially jeopardising data confidentiality and integrity, even though these functionalities are user-friendly for external users.

The VPCs would be able to get beyond internal network security measures like firewalls and Network Access manage Lists (NACLs), which are meant to protect the VPCs and manage access to internal resources, by using public APIs[4]. By allowing unauthorised access or malicious assaults to more easily obtain critical user profile data, this circumvention may erode the overall security posture. A more secure solution to these issues is to build internal APIs specifically designed to access user profile data inside of VPCs. The access to these internal APIs can be controlled by VPC-level security measures like security groups and

NACLs, and they can be secured with appropriate authentication mechanisms like access tokens or API keys.
Cupid maintains a regulated and secure access method for sensitive user profile data inside the VPCs through the usage of internal APIs. This method adds an extra degree of security by guaranteeing that access to the data is limited to authorised entities inside the VPCs. Additionally, Cupid may enforce compliance with data privacy rules, impose suitable encryption methods, and exert more granular control over access permissions—all of which are critical for preserving regulatory compliance—by using internal APIs.

Considering the sensitivity of user profile data, it is imperative to prioritize security and ensure that access mechanisms adhere to best practices. By embracing internal APIs and implementing robust security measures within the VPCs, Cupid can fortify the overall security of user profile data and alleviate potential risks associated with relying on public APIs for accessing sensitive information.

6. I wanted to use transit gateway to centrally route the traffic. However, since transit gateway cannot be connected to the on-premise network, I will have a mix of VPC peering and Direct Connect or VPN connection to the on-premise network.

Given the difficulties Transit Gateway presents in establishing direct connections to on-premise infrastructure, the suggested option of combining a combination of VPC peering, Direct Connect, or VPN connections to the on-premise network is a sensible and workable strategy. Alternative approaches are required because Transit Gateway cannot build a direct link to the on-premise network, despite offering a highly scalable and flexible service for optimising network connections and routing within the AWS architecture.

An essential part of this strategy is VPC peering, which permits private and secure communication between VPCs inside AWS regions. This technology makes it possible to exchange traffic across private IP addresses, which makes data transfer more effective and under control. Using VPC peering is essential to building an AWS network environment that works together.

It makes sense to employ Direct Connect or VPN connections to connect the on-premise network to the AWS infrastructure. By creating a dedicated, private network link between AWS and the on-premises infrastructure, Direct Connect guarantees communication security and dependability. In addition, VPN connections offer a secure route across the open internet by connecting AWS VPCs and the on-premise network via encrypted tunnels.

Using VPC peering, Direct Connect, or VPN connections together gives Cupid a strong hybrid communication configuration[5]. This guarantees effective communication between the on-premise system and the AWS Cloud across various regions, in addition to data transfer security. This hybrid solution fosters a smooth and integrated network environment that meets the unique connectivity requirements between on-premise and AWS resources, in line with Cupid's needs for flexibility, scalability, and security.

References

[1] "Using NAT Gateways with multiple-Amazon VPCs at scale | Amazon Web Services," *Amazon Web Services*, Sep. 12, 2023. https://aws.amazon.com/blogs/networking-and-content-delivery/using-nat-gateways-with-multiple-amazon-vpcs-at-scale/ (accessed Nov. 11, 2023).

[2] "SEC09-BP02 Enforce encryption in transit - Security Pillar," Amazon.com, 2023. https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/sec_protect_data_transit_encrypt.html (accessed Nov. 11, 2023).

[3] "Hub-and-spoke network architecture," *Google Cloud*, 2023. https://cloud.google.com/architecture/deploy-hub-spoke-vpc-network-topology (accessed Nov. 11, 2023).

[4] "How to use Amazon GuardDuty and AWS Web Application Firewall to automatically block suspicious hosts | Amazon Web Services," *Amazon Web Services*, Aug. 03, 2018. https://aws.amazon.com/blogs/security/how-to-use-amazon-guardduty-and-aws-web-application-firewall-to-automatically-block-suspicious-hosts/ (accessed Nov. 11, 2023).

[5] "Hybrid connectivity - Building a Scalable and Secure Multi-VPC AWS Network Infrastructure," *Amazon.com*, 2023. https://docs.aws.amazon.com/whitepapers/latest/building-scalable-secure-multi-vpc-network-infrastructure/hybrid-connectivity.html (accessed Nov. 11, 2023).