CSCI 5902 Advanced Cloud Architecting (Fall 2023)

Assignment 4 – Networking

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

Concerns:

a. Scalability: Deploying a NAT Gateway in every subnet across all VPCs globally might lead to scalability challenges. As the number of users grows in future, the number of subnets and the demand for outbound internet connectivity may increase significantly. This approach could become operationally complex and might not scale efficiently.

b. Cost Efficiency: NAT Gateways are charged $0.045 per hour and $0.045 per GB data processing [1]. Deploying them in every subnet across all VPCs globally can result in unnecessary costs, especially if certain subnets don't have high outbound traffic requirements.

Instead of deploying a NAT Gateway in every subnet, I would recommend a more centralized approach. Deploying a simple NAT Gateway in each VPC's public subnet would be more scalable, cost effective and also eases the overall network management burden. This central NAT Gateway would handle outbound internet connectivity for all private subnets within the same VPC. By using this way, we can reduce the number of NAT Gateways needed.

To achieve centralized outbound internet connectivity from private subnets in each region with a single NAT Gateway, we can follow these steps below [2]:

First, we need to create a public subnet with appropriate IP address range to host our NAT Gateway in each region. Next, create NAT Gateway in each public subnet and then associate it with new or existing Elastic IP addresses as needed. After that we have to update the route table of each private Amazon VPC subnet to direct outbound internet traffic to newly created NAT gateway. Also, enable Amazon CloudWatch to monitor NAT Gateways in order to ensure smooth functioning.

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.**

The decision to create a fully meshed VPC peering setup for all VPCs in each AWS region is a valid one. But it could lead to a lot of overhead, in connecting the networks. I would refer to establish VPC peering connections selectively, based on the specific communication needs between VPCs. This approach will reduce complexity and ensure that direct communication is only established where it's really necessary, making our network more efficient and manageable.

A completely interconnected VPC peering arrangement involves setting up peering connections among all VPCs within a specific region. This creates a network structure where each VPC is directly connected to every other VPC. In this configuration, each pair of VPCs is directly linked through a peering connection, resulting in a comprehensive mesh of connections [8].

To address the difficulties associated with the complexity and maintenance of a fully interconnected VPC peering setup, it is advised to selectively establish peering connections based on the specific communication needs between VPCs [8]. This involves creating peering connections only in cases where direct communication is essential. Through thoughtful design and planning of VPC peering connections, you can streamline the architecture, minimize maintenance tasks, and facilitate effective communication between the necessary VPCs.

AWS offers a feature known as VPC peering, allowing to link VPCs through the AWS network, facilitating communication between them using private IP addresses. This establishes a secure, private connection on a one-to-one basis between VPCs, eliminating the need for internet gateways, VPN connections, or extra hardware.

Setting Up VPC Peering:

To initiate VPC peering, create a peering connection between two VPCs and modify the route tables in each VPC to permit the exchange of traffic between them. It's essential that the VPCs either belong to distinct AWS accounts or reside in the same account but different AWS regions [7]. After establishing the peering connection, the VPCs can interact using private IP addresses, simulating a scenario where they are part of the same network [7].

VPC peering can simplify network architectures, enhance security, and reduce data transfer costs within the same region.

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

I would agree with Shay's decision. Using IPsec tunnels to encrypt all communication within VPCs is the best choice which ensures that the data is secured while at rest and in transit. It significantly boosts our security measures and ensures that we meet data privacy requirements. With IPsec, we can assure that our data is encrypted and safe from any unauthorized access as it moves through the network.

IPsec is a group of protocols for securing connections between devices [3]. IPsec (Internet Protocol Security), is a group of networking protocols utilized for encrypting IP packets, along with authenticating the source where the packets come from [3]. IPsec tunnels, also known as VPN (Virtual Private Network) tunnels, create a secure and encrypted connection between two endpoints over an untrusted network such as the internet [3]. In IPsec tunnel mode, the original IP header containing the final destination of the packet is encrypted, in addition to the packet payload [3]. So, IPsec tunnels provide encryption and authentication, ensuring the confidentiality and integrity of network traffic. They provide secure communication within VPC, or between VPCs or between VPCs and on-premises networks.

But setting up and managing IPsec tunnels can be complex, involving configuration and maintenance of VPN endpoints. IPSec can add overhead to your network and application traffic, hence the use of hardware appliances such as VPN Concentrators [4].

Overall, using IPsec tunnels in a VPC is an effective way ensures secure communication and protect sensitive user data while in transit.

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

I do not support the proposal of using the same Network Access Control Lists (NACLs) for all subnets. This approach might ease defining the security configuration but could lead to significant limitations and potential security risks. Network Access Control Lists (NACLs) play a crucial role in the AWS VPC (Virtual Private Cloud) network framework, serving as stateless firewalls for subnets. They manage inbound and outbound traffic at the subnet level, offering an additional layer of security compared to security groups, that operate in subnet level only. Utilizing distinct, unique NACLs is considered an optimal approach for enhanced network security.

Using the same NACLs for all subnets may not be ideal due to the following reasons:

a. Each and every subnet may have different security requirements based on the application hosted within it with varying security requirements. For instance, a subnet containing sensitive financial data requires huge restrictive access rules when compared to a subnet hosting publicly accessible applications.
b. Applying the same set of NACLs for every subnet, could lead to operational challenges as the infrastructure scales. Maintenance, troubleshooting, and adapting to changing security needs may become more complex. So, having separate NACLs for every subnet can provide better scalability, manageability and moreover, we can add, update or revoke rules set specific to each subnet without affecting other subnets.
c. By creating separate NACLs for each subnet, you can have granular control over the allowed traffic on comparing with one global subnet.

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

Using public APIs to access user profiles from VPCs may be convenient but it is not a good approach because it may lead to security breach. By using private APIs that are accessible only within the VPCs or through a secure VPN connection would be a better approach which I suggest. This ensures that user profile access is restricted to authorized as well as authenticated entities.

User profiles may contain sensitive information such as names, email addresses, and possibly even internal messages or photos. If any data leak happens then all the details in user profile will be exposed which often lead to unauthorized access and thus potentially violating user privacy. Moreover, public APIs are more vulnerable to DDoS attacks, or any other network attacks.

More secure approach I propose is use of private APIs or establish a secure VPN connection from VPCs

Private APIs are not exposed to the internet and their endpoints are only accessible from within the VPCs or through secure VPN connections. This restricts access to authorized and authenticated requests. Also, we can make use of API keys, OAuth 2.0, or token authentication. These methods will ensure our data is accessed by only authorized, authenticated clients.

In summary, relying on public APIs to access user profiles from VPCs poses a risk by exposing sensitive data to the public internet, thereby increasing security vulnerabilities. For a more secure approach, Shah should consider the use of private APIs or implement secure connections using API keys. This would enable better control over access and protect user profile data, mitigating the risk of unauthorized access and potential security threats.

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.**

Transit gateway along with the other given options would be a liable as it cannot be connected directly to cloud from the on-premise network. Using a combination of VPC peering and either Direct Connect or VPN connections provides connectivity between the on-premise infrastructure and the AWS Cloud.

Transit gateway simplifies network management by providing a central hub for routing traffic between connected VPCs and on-premise networks. It streamlines connectivity thus reduces the need for complex VPC-to-VPC peering configurations [5]. It can handle a large number of VPCs and connections, making it suitable for dynamic and expansive architectures.

Why Shah's decision is right: The combination of VPC peering, Direct Connect, or VPN connection provides multiple options for establishing secure communication between the on-premises infrastructure and AWS Cloud. It allows for the secure transfer of sensitive user data in compliance with data privacy laws.

But setup might require more configuration and management as it involves multiple components (VPC peering, Direct Connect, or VPN) and the performance of VPN connections may not match that of Direct Connect, especially for largescale data transfers.

Shah might have considered using AWS Transit Gateway and combining it with AWS Direct Connect Gateway as an alternative. Direct Connect Gateway allows us to connect Direct Connect links to multiple VPCs across different regions, providing a centralized on-premise connection. With AWS Direct Connect SiteLink, we can send data between AWS Direct Connect locations to create private network connections between the offices and data centers in your global network [6]. This might simplify the connectivity model and reduce the need for a mix of connections.

References

[1] *Amazon.com*. [Online]. Available: https://aws.amazon.com/vpc/pricing/. [Accessed: 17-Nov-2023].

[2] A. W. S. Official, “How do I set up a NAT gateway for a private subnet in Amazon VPC?,” *Amazon Web Services, Inc*, 29-Aug-2018. [Online]. Available: https://repost.aws/knowledge-center/nat-gateway-vpc-private-subnet. [Accessed: 17-Nov-2023].

[3] *Cloudflare.com*. [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-ipsec/#:~:text=IPsec%20tunnel%20mode%20is%20used,addition%20to%20the%20packet%20payload. [Accessed: 17-Nov-2023].

[4] J. Turnbull, “The pros and cons of IPsec,” *Data Center*, 28-Oct-2005. [Online]. Available: https://www.techtarget.com/searchdatacenter/answer/The-pros-and-cons-of-IPsec. [Accessed: 17-Nov-2023].

[5] *Amazon.com*. [Online]. Available: https://aws.amazon.com/transit-gateway/. [Accessed: 18-Nov-2023].

[6] *Amazon.com*. [Online]. Available: https://aws.amazon.com/directconnect/. [Accessed: 18-Nov-2023].

[7] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/peering/create-vpc-peering-connection.html. [Accessed: 18-Nov-2023].

[8] “Examples of VPC peering connections - Virtual Private Cloud - Alibaba Cloud Documentation Center,” *Alibabacloud.com*. [Online]. Available: https://www.alibabacloud.com/help/en/vpc/user-guide/examples-of-vpc-peering-connections. [Accessed: 18-Nov-2023].