Introduction

SDP is designed around the user identity, not the IP address. IP addresses exhibit too much implicit trust. SDP is also called black out, which means that sensitive data and applications cannot be detected by unauthorized users. In this module, we will start with an overview of an SDP architecture that closely follows the Cloud Security Alliance guidelines. The architecture includes the SDP components, such as an SDP controller, SDP gateway, initiating hosts, and accepting hosts, and use a lightweight security protocol called Single Packet Authorization. We will follow this discussion into the primary functions of SDP, such as information and infrastructure hiding, and micro-segmentation. We will further highlight what advantages these functions bring to organizations. We will end this module comparing traditional VPN access to SDP. Here, you will see the true benefits of implementing SDP in conjunction or by replacing your existing VPN architecture.

SDP Architecture

The Cloud Security Alliance has proposed a number of SDP architectures that you can follow. The primary components of an SDP architecture include a client initiating host, a service accepting host, SDP gateways, and an SDP controller, to which the accepting hosts and the initiating hosts both connect to. All of the components can be redundant for scale and uptime. The SDP controller, which is the policy engine and central authentication point, a policy's assigned to the client, which determines its allowed resources. The controller provides the control there, and allows initiating and accepting hosts to be authenticated prior to allowing communication. Both the controller and accepting host are protected by SPA, Single Packet Authorization, making them invisible and inaccessible to unauthorized users and devices. Everything else is dark, hidden from the user. As it's the central authentication and policy engine, it is typically connected to an organization's identity access management and public key infrastructure. The next component is the SDP gateway. The gateway sits in front of the resources, protecting them. It provides authorized users and devices with access to protected processes and services. Once the policy is granted by the controller, you could say the gateway opens the door to the network, and closes it very quickly again, back to a default in deny. Keep in mind, there is no inbound connectivity to the gateways, or application ports open. This hides and protects all applications that are located behind the gateway. SDP host can either initiate connections, or accept connections. These actions are managed by interactions with the SDP controllers, via the control plane. The client that consists of the SPA packet, with the authentication and message keys, the devices can be a computer or a mobile device, all of which are user facing, meaning the SDP client software is run on the device. Once the controllers are added to the SDP domain, they can be connected to a variety of authentication and authorization services, such as PKI service, device attestation, geolocation, and other similar services. The accepting hosts are then added to the SDP, and connect and authenticate to the controller. The accepting hosts do not acknowledge communication from any other host, and will not respond to any non-provisioned requests. When an initiating host connects to the network, they authenticate to the controller. The controller will then carry out some initial policy verifications to validate the resources that the users have authorization for. After authenticating initiating host, the SDP controller determines a list of authentication hosts to which the initiating host is authorized to communicate. The SDP controller instructs the accepting host to accept communication from the initiation hosts, and initiate any optional policies required for encrypted communications. The controller is essentially setting up the control plane for data plane activity to occur. The initiating host initiates a SPA packet to each authorized authenticated host. The initiating host then creates a two-way encrypted connection to those accepting hosts. The two-way encrypted connections are carried out with mutual TLS. The usual configuration for TLS is that clients authenticate the servers, but mutual TLS ensures that both parties are authenticated. This means that only validated devices and users can become authorized members of the SDP. The gateways are dynamic. They can adjust user access based on, for example, user location, and ask for a higher authentication method if the user is deemed to be in a risky location. This is also known as step-up authentication, whereas under certain circumstances, for example prompting for an OTP in order to access sensitive systems. All SPD protected servers and services are located within the SDP, and are hidden from all unauthorized network queries and scan attempts. Then, real-time encrypted connections are created between the requesting client and the protected asset. SDP requires clients to be authenticated and authorized before obtaining access to protected assets, regardless of whether the connection is initiated inside or outside of the network perimeter. An example of this SDP architecture consists of five main features; single-packet authorization, mutual transport layer security, device validation, dynamic firewalls, and application binding; each of which provides a number of reasons why you would implement SDP.

Information and Infrastructure Hiding

We all know that you cannot protect what you cannot see, but you also cannot attack what you cannot see. Firstly, information and infrastructure hiding. When you walk into a house, you can enter every room. Whether you own the house or the contents of each of the rooms, you are free to move around and touch someone else's belongings. Maybe it's a valuable piece of crystal, or something that's not so valuable, such as a deck of playing cards. The fact is, you can touch whatever you want, even if it's not yours. Zero trust SDP changes all of this, and you cannot even see what is not yours, let alone touch. For the purposes of information infrastructure hiding, SDP uses the concept of black out, which means that sensitive data cannot be detected by unauthorized users. SDP creates a virtualized network segment that abstracts applications and data from the underlying physical infrastructure. Access is restricted to authenticated users who could only see those resources they're authorized to access. All other segments are hidden from view, so they cannot be compromised. SDP's new approach to networking and security can be used for information infrastructure hiding through the use of a lightweight protocol called Single Packet Authorization. SDP's prime objective is to make applications and infrastructure completely undetectable; no internal IP addressing or DNS information is shown, creating an invisible network. SDP provides zero visibility and zero connectivity, only establishing connectivity after clients prove they can be trusted to allow legitimate traffic. Protected assets can be hidden regardless of location, be it on-prem, public, or in private clouds, a DMZ, or simply a server on an internal LAN. This keeps with today's hybrid environments. The base for network security started by limiting visibility. You cannot attack what you cannot see. Networks are separated from public and private address ranges. If a host was assigned a public address and wanted to communicate with a host that was configured with a private address, it would need to go to another device, and have a permit policy set. Limiting visibility this way works to a degree, but we cannot get away from the fact that A, if you have the IP address of someone, you can connect to them; B, if a port is open, you can potentially connect to that too. Therefore, the traditional method of security can leave your network wide open for compromise, especially when actors have all the tools at hand. Essentially, SDP provides a blackened network

The protected servers will not respond to any connection until the clients have successfully authenticated and authorized. This approach mitigates denial of service attacks. Anything that is internet-facing and reachable on the public internet is therefore susceptible to these types of attacks in the traditional network connectivity construct, but SDP mitigates these attacks because it discards all bad packets and only accepts good packets. The default-drop firewall is deployed with no visible presence to unauthorized users, only good packets are allowed through. This also allows for attack detection. If a host receives anything other than a valid SPA packet or a similar construct, it views that packet as being part of a threat. The first packet to service must be a valid SPA packet. If it receives any other type of packet, it views this as an attack, which is useful for bad packet detection. Therefore, SDP is able to determine an attack based on a single malicious packet, which is a highly effective way of detecting network-based attacks. All external network and cross-domain attacks can be detected.

Mutual Encrypted Connections

Secondly, mutually encrypted connections. The Software-Defined Perimeter requires mutual two-way cryptographic authentication, often carried out with mutual TLS. SDP uses the full TLS standard to provide mutual two-way cryptographic authentication. Mutual TLS provides this, and goes one step further to authenticate the client. Mutual TLS connections are set up between all components in the SDP architecture. The connections between all components must use mutual authentication for validation, as authorized members of the SDP prior to further device validation and user authentication. Transport layer security was designed to provide mutual device authentication prior to enabling confidential communication over the public internet; however, the common TLS configuration is the validation that ensures that the client is connected to a trusted entity. So the typical TLS adoptions are to authenticate servers to clients, not clients to servers. The typical TLS configuration ensures that authentication doesn't happen the other way around in order to validate that the remote entity knows it's communicating with a trusted client. Firstly, this offers a robust device and user authentication, as connections from unauthorized users and devices are mitigated. Secondly, forged certificates, which are attacks aimed at credential theft are disallowed. Mutual authentication schemes pin certificates to a known and trusted valid root managed by the SDP, and will not consist of hundreds of root certificates trusted by most consumer browsers. This will mitigate against impersonation attacks, where a bad actor can forge a certificate from a compromised certificate authority. In a publicized attack against a certificate authority, the bad actor took complete control over all eight of the certificate issuing servers in which they issue rogue certificates. At that time, it was estimated over 300, 000 users had their private data exposed by rogue certificates. As a result, their certificate issued by the CA were immediately blacklisted by browsers. Finally, man-in-the-middle attacks are disallowed. These types of attacks use forged or obsolete online certificate status protocol response stapling. The mutual handshake technology protects from man-in-the-middle attacks that exploit OSCP responses before the server certificate is revoked.

Need-to-know Access Model

Thirdly, SDP employees a need to know access model. SDP permits the requesting client to view only the allowed resources appropriate to the assigned policy. As you may recall from module one, the policy has a number of facets; user access policy that touches on authorization, network access policy touches on what to connect to, and user account policies touches on authentication. With either one, there is no policy visibility with IP addresses. Users are associated with their devices that are validated based upon policy. Only connections to the specifically requested service are enabled, and no other connection is allowed to any other service. Access is granted directly between the user and their devices to the application and resource. There is simply no concept of inside and outside of the network. This removes the network location point as a position of an advantage, and eliminates the excessive implicit trust that IP addresses exhibit. Connectivity is based on a need-to-know access model, which enforces the zero trust principles of least privilege. With traditional networks establishing who made the connection from what device to what service is very complicated to have full visibility into all established connections. Firewall logs and IDS logs must be streamed in real-time to a log aggregator in order to establish where the connection is originating. But the origin of the connection is typically obscure because of \_\_\_\_\_. Moreover, it takes days, and sometimes months to gather these logs into a single aggregator to recreate the connection information. Network switch and router logs are sent to a syslog server, and then to the aggregator in order to establish where the connection is ended. Logs from Active Directory or similar authentication servers will need to be sent to the aggregator in order to establish who made the connection. A separate device management scheme is implemented to establish authorized devices adding to this complication. SDP provides additional information, such as who made the connection, from what device, and to what service. This gives you full visibility into all the established connections, something that is pretty hard to do if you have an IP-based solution. This makes forensics easier. The SDP only accepts good packets, and any bad packets can be analyzed and tracked for forensic activity. Secondly, it enforces device validation, which helps against threats from unauthorized devices. The mutual two-way encrypted communications prove that the requested device possesses a private key that has not expired or been revoked. This ensures that the private key is held by the proper device. Also, device validation ensures that the device is running trusted hardware and is being used by the appropriate user. Thirdly, it mitigates credential theft. The objective of device validation is to prove that the proper device holds the private key, and the software running on the device can be trusted. Keys are proved to be held by the proper device that is requesting the connectivity, thereby eliminating credential theft. There is complete lockdown on lateral movements, as a user is only allowed access to the resources it is authorized to, and everything else is dark. Because all connections are known, you can drop a connection instantly, incident response in a matter of seconds. You can't do that today because you have to gather all the logs and recreate all the connection information in order for you to drop a connection, but by that time, the damage is already done.

Dynamic Access Control

SDP deploys a dynamic firewall that starts off with one rule, deny all. Request to communication is dynamically inserted into the firewall providing dynamic firewall security policy as opposed to static configurations. Every packet hitting the firewall is inspected for SPA, then is quickly verified by the SDP for a connection request. Once established and the firewall is closed again, connections made are not seen by rogue outsiders or rouge insiders within the network. Traditional firewalls use static configuration to limit incoming and outgoing requests based on the information in the IP packet. This is a source IP address or port, or destination IP address and port, this type of firewall is limited in scope, as they cannot express or enforce rules based on identity information, making zero trust very difficult to implement. Also, an abundance of firewall rules introduces a lot of complexity. Attempting to model identity centric control with the limitations of five tuple is difficult. Five tuple is limited, as you can base policy only on a set of five different values that comprise of TCP IP connection. SDP can be used alongside traditional firewalls, and can take over the network access control enforcement that we attempt to do with traditional firewalls. It also allows dynamic membership-based enclaves that prevent network-based attacks. The SDP dynamically binds users to devices, and then dynamically enables those users to access protected resources by dynamically creating and removing firewall rules. Access to protected resources is enabled by dynamically creating and removing inbound and outbound access rules. SDP enables richer and more precise access control mechanisms, and will reduce the firewall rule set considerably.

Identity-driven Access Control

The SDP is based on identity-driven access control, which means that access controls are based upon strong authentication, end-to-end encryption, and the trust relationship between source and destination systems.

to authorize applications while remaining blacklisted from unauthorized applications. SDP secures connections, and is a connection-based paradigm that is independent of the IP infrastructure. There are several connection-based models as defined by the CSA. We have client-to-gateway, client-to-server, server-to-server, client-to-server-to-client, and client-to- gateway, which will be discussed later. The client-to-gateway and gateway-to- gateway connection model closely resemble how traditional networks are set up today. However, because SDP is designed to have a control channel and a separate data channel, users gateways and service initiating connections are authenticated first before the required connections are established providing access. Regardless of the connectivity model, SDP does not allow access until initiating clients, or gateways, or servers authenticate first, and then are verified for authorize prior to access. Identity-driven access control can be identities for users and non-persons identities. Traditional network solutions provide coarse grained network segmentation based on someone's IP address; however, someone's IP address is not a valid hook for security, and does not provide much information about the users identity. Also, access was provided first, and then authentication and authorization are enforced. This requires knowledge of threats, bad IP lists, AI-based validation, and other expensive and complicated mechanisms to figure out who to provide access to. This is the root of the problems in traditional networks. SDP allows the creation of identity centric access controls, and is a simple method for establishing zero trust. For example, SDP can permit a policy that only allows finance users web application access to the financial management system, but only from corporate managed devices, and not from any other type of device. SDP can also go a layer deeper, and ensure that access is only allowed when specific conditions are met, for example, during weekdays, and from certain locations. Access can then be allowed for compliance reporting.

Application Layer Access

SDP only grants access to the specific applications and an application layer. SDP has complete control of which devices and applications are permitted to access what services at the application layer, permitted by a policy granted by the SDP, devices can only access specific hosts and services, and cannot access any network segments or subnets that it's not entitled to. Therefore, broad network access is eliminated, reducing the tax service to an absolute minimum. SDP solutions may white list applications on the user's device. SDP can be used to protect applications, as well as system level resources. After authenticating and authorizing both the device and the user, the SDP creates two-way encrypted communications to protected applications. Traditional applications were grouped into VLANs, whether they offered similar services or not. Everything on that VLAN was reachable. This complicates implementation of zero trust. SDP provides fully encrypted application communication path. The application binding permits only authorized applications, so they can only communicate through the established tunnels, thus blocking all other applications from those tunnels. This creates a dynamic perimeter around the application, which includes the users and devices that connect to it. This offers a very no access path, that is, a single connection vetted every time one is created.

Microsegmentation

Network micro-segmentation is the practice of dividing the network into smaller logical segments, so that only authorized endpoints can access the applications and data housed in those segments. This provides many benefits over traditional perimeter security as the smaller segments present or reduce a tax service for malicious actors. SDP enables a creation of micro- segmentation that are based on the user defined controls enabling a 1-to-1 mapping. Unlike with a VLAN, where there is the potential to see everything on that VLAN. Micro-segmentations are based on user- defined controls enabling a 1-to-1 mapping. Therefore, if a bad actor gains access to one segment in the zone, they are prevented from compromising any other network resource in that zone. This reduces attack surface to an absolute minimum, by reducing the number of devices that can be communicated on any given segment.

Issues with Traditional VPN

VPN's establish secure, private connections across untrusted networks, enabling secure remote access. While VPNs do provide the encapsulation and encryption of network traffic, they have many limitations that are better addressed with SDP. They do not secure users when they land on a network segment, and are based on coarse grain access control, where the user's access to entire network segments and subnets. This violates the main zero trust principle of least privilege. VPNs generally provide broad overly permissive network access with only basic access control limits that are based on subnet ranges. Traditional VPNs introduce a lot of complexity. What do you do when you've got multiple sites that users need to access applications in? In this type of scenario, the cost of management would be very high. Users would have to make many changes to their VPN client software in order to gain access to the applications in different locations. Traditional VPNs are complex for the administrators to manage, and for the users to operate. Users are required to remember which applications require the use of the VPN, and which applications do not. They also have to manually connect and disconnect. That poor user experience is most likely at the surface, as you need to back all the user traffic to a regional datacenter, and have the user choose from a list of VPN gateways that are used for the different applications. This adds latency, bandwidth cuts, and also a management overhead. Finally, the VPN server itself is exposed as a service on the internet, making it invisible and vulnerable to attack.

SDP vs. VPN

In this demo, we're going to display how users access applications by traditional means, and with SDP. With the traditional way, a client enters an on-premise datacenter or cloud to access an application by creating a tunnel known as a virtual private network. The tunnel enables the forwarding of traffic from the client's device to the application location. What's really happening here is that the tunnel creates an extension between a client's device and the application location. The tunnel is based on IP addresses, which are set on the client's device, and the remote application endpoint. The IP connectivity between a client and the application of the network where the application's located is extended to the client. However, the client may be sitting in an insecure location, for example, internet café, or a hotel room, all of which may not be sufficiently protected, and such locations should be considered insecure. From a security standpoint, the traditional method of VPN access enables clients to have a broad network level access, and makes the network susceptible to lateral movements. On the other hand, with SDP, the client connects to the SDP, which carries out security checks and forwards the request to the application. SDP still allows clients to access the application, but they're only allowed to access that specific application and nothing more. Everything else is dark and hidden. The users cannot get to any other application. Clients need to pass certain security patrols and there is no broad level access that is susceptible to lateral movements, and access is controlled based on a white list ruleset instead of the traditional black list ruleset. Also, instead of using IP addresses as a client identifier, other variables are used. As a result, the application is now the access path and not the network. SDP is purposely designed for fine-grained access control, whereby all unauthorized resources are inaccessible to users, which adheres to the zero trust principles of least privilege. It secures both the remote and on-premise users with a holistic solution that enables you to retire your existing VPN solution, or run in parallel. By using SDP, organizations can have a single access control platform that is consistent for remote, on-premise, and mobile device users.