République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Blida 1
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Sécurité des Systèmes Informatiques

Thème

Conception et Implémentation du Software Defined Perimeter (SDP) pour la mitigation de l'Attack Denial of Service

Organisme d'accueil : **SONATRACH**

Réalisé par :
Mr Seffar Mehdi
Mr Diab Mehdi

Encadreur : Mr Mekideche Mounir
Co-encadreur : Mr Benhassine Oualid
Promoteur : Mr BENYAHIA Mohammed

Devant le jury :
Président : Mme OUKID Saliha
Examinateur : Mr. OULD KHAOUA Mohamed
Promoteur : Mr. BENYAHIA Mohammed

Promotion 2020/2021

Acknowledgment

First of all, we want to thank Allah, the Almighty, for giving us the strength and patience to complete this work.

Then, we would like to express our warmest thanks and gratitude to our mentor Benhassine Oualid for his continuous supervision, for constructive remarks and for his valuable advice throughout the period of our work.

We would like to thank the members of the jury for taking part in the evaluation of this work.

A few people have contributed to this work and deserve thanks.

We are also indebted to our promoter Benyahia Mohammed, who believed in our ability to achieve something great. By giving us the opportunity to pursue this dream and make it come true. With a heart full of gratitude.

Finally, we want to thank our families for their encouragement, their help and their great patience with us.

Abstract

Open ports on a service can be a threat since intruders can scan them for important information. Granting only authorized users access to these ports would safeguard a service from hackers collecting intelligence about it before attacking.

The Single Packet Authentication (SPA) mechanism solves that problem by allowing clients to authenticate themselves without using a connection.

The Software Defined Perimeter (SDP) is a protocol that utilizes SPA to enable secured access to protected services that are secured behind a Black Cloud barrier.

The goal of this project is to provide SPA and SDP protocol implementations, demonstrate their effectiveness against port scanning and Denial of service attacks and present/discuss their architecture.

Keywords:

Zero-Trust security, Software defined perimeter, Single Packet Authorization, Denial of service attack, Cryptography, access control, Port scanning

Resumé

Les ports ouverts sur un service peuvent être une vulnérabilité car les intrus peuvent les analyser àla recherche d'informations importantes. Accorder uniquement aux utilisateurs autorisés l'accès à ces ports protégerait un service contre les pirates qui collectent des informations à son sujet avant d'attaquer.

Le Single Packet Authentication (SPA) résout ce problème en permettant aux clients de s'authentifier sans utiliser de connexion. À l'aide du Software Defined Perimeter (SDP), un protocole qui utilise le SPA afin de fournir un accès à des services protégés qui sont sécurisés derrière une construction appelée "Black Cloud".

L'objectif de ce projet est de fournir des implémentations de protocoles SPA et SDP, de démontrer leur efficacité contre l'analyse des ports et L'attaque par déni de service et de présenter/discuter leur architecture.

Mots clés :

Sécurité Zero-Trust, Software defined perimeter, Single Packet Authorization, Attaque par déni de service, Cryptographie, Contrôle d'accès, Analyse de port.

ملخص

يمكن أن تكون المنافذ المفتوحة على الخدمة ثغرة أمنية حيث يمكن للمهاجمين الحصول على معلومات قيمة عن طريق تفحصها. إن السماح للمستخدمين المصرح لهم فقط بالوصول إلى هذه المنافذ من شأنه حماية الخدمة من المتسللين الذين يحاولون جمع المعلومات عنها قبل الهجوم عليها.

تعمل آلية The Single Packet Authentication على حل هذه المشكلة عن طريق السماح للعملاء لمصادقة أنفسهم دون استخدام اتصال.

المحيط المحدد بالبرمجيات (SDP) هو بروتوكول يستخدم SPA لتمكين الوصول الآمن إلى الخدمات المحمية المؤمنة خلف حاجز السحابة سوداء Black Cloud.

الهدف من هذا المشروع هو توفير تطبيقات بروتوكول SPA و SDP ، وإثبات فعاليتها ضد مسح المنافذ وهجمات رفض الخدمة وتقديم / مناقشة بنيتها.

الكلمات الرئيسية: أمان الثقة المعدومة ، Software defined perimeter , Single Packet Authorization, هجوم رفض الخدمة, التشفير,التحكم في الوصول, فحص المنافذ.

Table of contents

-

Figures list

Tables list

Acronym list

WHO	World Health Organization
OPTK	One Time Port Knocking
SPA	Single packet authorization
SDP	Software Defined Perimeter
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
FTP	File Transfer Protocol
SSH	Secure Shell
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
Nmap	Network mapper
ARP	Address Resolution Protocol
DDoS	Distributed Denial-of-
ServiceVoIP	Voice over IP
SMTP	Simple Mail Transfer
ProtocolSaaS	Software as a Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
SYN	Synchronize
GPS	Global Positioning
SystemRSA	Rural Service Area
DES	data encryption standard
AES	Advanced Encryption Standard
EEC	Elliptic Curve Cryptography
DSS	Digital Signature Standard
MD5	Message-Digest algorithm 5
SHA-3	Secure Hash Algorithm 3
HMAC	Hash-Based Message Authentication Codes
CSR	Certificate signing request (CSR)

Introduction

General introduction

On January 2020, the World Health Organization (WHO), announced officially the public health emergency due to the spread of corona viruses who has spread to almost every country in the world threating not only human lives but also economy, business and education. forcing the world to change.

One of the most initial impacts of COVID-19 is shifting from the physical workplace to the online virtual workplace, that fact motivated firms, companies and organization all around the world to quickly make their services accessible via the internet, and the most important challenge was to maintain security in all that rush.

Organizations and firms had to deal with the growing security demands emerging from the risk of making almost everything accessible via the internet, as known by security specialists, the CIA triad comprising of Confidentiality, Integrity and Availability is the heart of Information Security [2]. so those three-constraint had to be respected by all organizations aiming to be present online. Otherwise, this could negatively affect their reputation, the daily-life of their client and workers or even cause the loss of huge amount of money. For example, the popular Zoom got a really bad reputation because of security professionals, advocates and even the FBI who warns that his default settings are not safe. As a result, many companies such as NASA, SpaceX, and countries, including Taiwan, USA, and the Australian Defence force, banned Zoom for communication [1].

To secure the processes of making services and data available online, organizations have to choose a solution which is easy to monitor, quickly implementable in real life and easily scalable. Offering a good access control and protection from attacks that can occur from the inside or the outside. As a result, an exploration into new security measures to protect online available services has commenced because traditional perimeter techniques have proven to be inadequate in protecting the infrastructure from network attacks [3, 4].

In this project, to achieve the security solution that can meet the requirement for the need of those organizations we are using the software-defined-perimeter because it adopts a need-to know model where the device's/application's identity is granted access to the application infrastructure upon first being verified and authenticated [3,4]. Due to this selective process, the infrastructure is referred to as "black." This is because it cannot be detected by users who are unauthorized to see it [3,4]. this can effectively mitigate many network-based attacks including server scanning, denial of service, and man-in-the middle
attacks, among many others [3,4].

Our Objective

Sonatrach enterprise want to make their services and data accessible to employees remotely, that's why they suggested us a new solution that is easy to monitor, inexpensive, scalable and secured. It is Software Defined Perimeter (SDP).

To implement this solution, we have to:

- Make all services and data connected to the internet.
- Restrict access to all services and data using gateways and firewalls making them go dark.
- Implement a control party that monitor services and data accessibility.
- Give access to employees to see the concerned services.
- Identify and authenticate employees.
- Create and secure the connection between employees and services.

The Plan

Our work is structured to four chapters:

- **Chapter 1:** Common network attacks, cryptography and defense mechanisms.
- **Chapter 2:** Authentication prior to connection.
- **Chapter 3:** Software Defined Perimeter.
- **Chapter 4:** Implementation of Software Defined Perimeter.

We will complete our research with a conclusion and perspectives of future work.

Chapter 1: Common network attacks, cryptography and defense mechanisms

1.1- Introduction

Nowadays, more than ever, business and the society as a whole are linked to computer technology in multiple ways, and are therefore also strongly affected by its malfunctions and security breaches. Poor security practices may cost firms huge amounts of money [8]. They can also have a severe impact on human activities or even human lives.
In this chapter we will explain some common network attacks.

1.2- Common Network Attacks

The main target of this project is to build a reliable security system that provides mechanisms to minimize the risk of attacks. In the following we briefly describe the commonly used network attacks.

• **Sniffing** [9]:

Packet sniffing is a method of tapping each packet as it flows across the network; i.e., it is a technique in which a user sniffs data belonging to other users of the network. Packet sniffers can operate as an administrative tool or for malicious purposes. It depends on the user's intent. Network administrators use them for monitoring and validating network traffic. Packet sniffers are basically applications. They are programs used to read packets that travel across the network layer of the Transmission Control Protocol/Internet Protocol (TCP/IP) layer. (Basically, the packets are retrieved from the network layer and the data is interpreted.) Packet sniffers are utilities that can be efficiently used for network administration. At the same time, it can also be used for nefarious activities. However, a user can employ a number of techniques to detect sniffers on the network and protect the data from sniffers. Sniffing Attacks can be prevented by using strong cryptography mechanisms in order to obfuscate the transmitted data (Symmetric or asymmetric cryptography such as RSA, DES, AES etc.).

• **Port Scanning** [10]:

Port scanning is a method of determining which ports on a network are open and could be receiving or sending data. It is also a process for sending packets to specific ports on a host and analysing responses to identify vulnerabilities. This scanning can't take place without first identifying a list of active hosts and mapping those hosts to their IP addresses. This activity, called host discovery, starts by doing a network scan. The goal behind port and network scanning is to identify the organization of IP addresses, hosts, and ports to properly determine open or vulnerable server locations and diagnose security levels. Both network and port scanning can reveal the presence of security measures in place such as a firewall between the server and the user's device. After a thorough network scan is complete and a list of active hosts is compiled, port scanning can take place to identify open ports on a network that may enable unauthorized access. It's important to note that network and port scanning can be used by both IT administrators and cybercriminals to verify or check the security policies of a network and identify vulnerabilities — and in the attackers' case, to exploit any potential weak entry points. In fact, the host discovery element in network scanning is often the first step used by attackers before they execute an attack.

As both scans continue to be used as key tools for attackers, the results of network and port scanning can provide important indications of network security levels for IT administrators trying to keep networks safe from attacks.

Port numbers:

Computer ports are the central docking point for the flow of information from a program or the Internet, to a device or another computer in the network and vice versa, a parking spot for data to be exchanged through electronic, software, or programming-related mechanisms. The port number combined with an IP address form the vital information kept by every Internet Service Provider in order to fulfil requests. Ports range from 0 to 65,536 and basically rank by popularity. Ports 0 to 1023 are well known port numbers that are designed for Internet use although they can have specialized purposes as well. They are administered by the Internet Assigned Numbers Authority (IANA). Here are the most prominent ports and their assigned services:

Port 20 (UDP) holds File Transfer Protocol (FTP) used for data transfer.

Port 22 (TCP) holds Secure Shell (SSH) protocol for secure logins, ftp, and port forwarding.

Port 53 (UDP) is the Domain Name System (DNS) which translates names to IP addresses.

Port 80 (TCP) is the World Wide Web HTTP.

Numbers 1024 through 49151 are considered “registered ports” meaning they are registered by software corporations. Ports 49,151 through 65,536 are dynamic and private ports - and can be used by nearly everyone.

Port scanning technique:

They differ depending on the specific goal or attacker strategy.

- Ping scans: The simplest port scans are called ping scans. In a network, a ping is used to verify whether or not a network data packet can be distributed to an IP address without errors. Ping scans are internet control message protocol (ICMP) requests and send out an automated blast of several ICMP requests to different servers to bait responses. IT administrators may use this technique to troubleshoot, or disable the ping scan by using a firewall — which makes it impossible for attackers to find the network through pings.

- Half-open or SYN scans: A half-open scan, or SYN (short for synchronize) scan, is a tactic that attackers use to determine the status of a port without establishing a full connection. This scan only sends a SYN message and doesn’t complete the connection, leaving the target hanging. It’s a quick and sneaky technique aimed at finding potential open ports on target devices.

- XMAS scans: XMAS scans are even quieter and less noticeable by firewalls. For example, FIN packets are usually sent from server or client to terminate a connection after establishing a TCP 3-way handshake and successful transfer of data and this is indicated through a message “no more data is available from the sender.” FIN packets often go unnoticed by firewalls because SYN packets are primarily being looked for. For this reason, XMAS scans send packets with all of the flags — including FIN — expecting no response, which would mean the port is open. If the port is closed, a RST response would be received. The XMAS scan rarely shows up in monitoring logs and is simply a sneakier way to learn about a network’s protection and firewall.

Port scan results:

Port scan results reveal the status of the network or server and can be described in one of three categories: open, closed, or filtered.

- Open ports: Open ports indicate that the target server or network is actively accepting connections or datagrams and has responded with a packet that indicates it is listening. It also indicates that the service used for the scan (typically TCP or UDP) is in use as well. Finding open ports is typically the overall goal of port scanning and a victory for a cybercriminal looking for an attack avenue. The challenge for IT administrators is trying to barricade open ports by installing firewalls to protect them without limiting access for legitimate users.

- Closed ports: Closed ports indicate that the server or network received the request, but there is no service “listening” on that port. A closed port is still accessible and can be useful in showing that a host is on an IP address. IT administrators should still monitor closed ports as they could change to an open status and potentially create vulnerabilities. IT administrators should consider blocking closed ports with a firewall, where they would then become “filtered” ports.

- Filtered ports: Filtered ports indicate that a request packet was sent, but the host did not respond and is not listening. This usually means that a request packet was filtered out and/or blocked by a firewall. If packets do not reach their target location, attackers cannot find out more information. Filtered ports often respond with error messages reading “destination unreachable” or “communication prohibited.”

Nmap:

Nmap is a free open-source tool, employed to discover hosts and services on a computer network by sending packets and analyzing the retrieved responses. It offers some features for probing computer networks, including host discovery and service and operating system detection.

- Nmap can provide further information on targets, including reverse DNS names, device types, and MAC addresses.
- Host discovery, identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.

- Port scanning, Enumerating the open ports on target hosts.
- OS detection – Determining the operating system and hardware characteristics of network devices.
- Version detection – Interrogating network services on remote devices to determine the application name and version number.
- Scriptable interaction with the target support using the Nmap Scripting Engine (NSE).

• **Identity Spoofing** [11]:

Spoofing is the act of disguising a communication from an unknown source as being from a known, trusted source. Spoofing can apply to emails, phone calls, and websites, or can be more technical, such as a computer spoofing an IP address, Address Resolution Protocol (ARP), or Domain Name System (DNS) server. Spoofing can be used to gain access to a target's personal information, spread malware through infected links or attachments, bypass network access controls, or redistribute traffic to conduct a denial-of-service attack. Spoofing is often the way a bad actor gains access in order to execute a larger cyber-attack such as an advanced persistent threat or a man-in-the-middle attack.

Types of Spoofing :

1. ARP Spoofing:

ARP spoofing occurs by binding the spoofer's MAC address (their Media Access Control address) to a legitimate IP address's default local access network (LAN) gateway. Essentially, a spoofer takes the place of the destination IP and through that spoofing, gains access to their local network. With this access, they capture sensitive information and access unrestricted information on the network. They also manipulate information before it reaches the legitimate IP address. Spoofers then carry out phishing and pharming attacks and assume new identities based on the information they receive. Additionally, ARP spoofers attempt a distributed denial-of-service attack (DDoS) which overwhelms existing security systems by dramatically increasing the number of users it must authenticate.

2. MAC Spoofing

Each device should have a unique Media Access Control address (MAC) that should not be encountered elsewhere. However, spoofers take advantage of vulnerabilities and imperfections in hardware to spoof the MAC address. As a result, the local network recognizes the MAC address and bypasses certain security protocols. Because spoofers operate with a trusted address, other users fall victim to business email compromise fraud, data breaches, and more. In addition, with trusted access, a spoofed address can deposit malware on a local network. Spoofers then prey on vulnerabilities and steal sensitive information.

3. IP Spoofing

The source or destination of a virtual message traces back to an IP address associated with a physical location. However, spoofers mask themselves with a legitimate IP address or assume the IP address of someone in that low-risk geolocation. Because many systems do not implement authentication protocols, the masked IP address takes the place of the legitimate source without the legitimate sender or recipient's knowledge. With this IP spoof, a spoofer can deploy a man-in-the-middle attack within a network, allowing them to steal sensitive information and inform themselves for future fraud attempts. IP spoofing relates to geolocation spoofing:

4. Geolocation Spoofing

One can spoof their geolocation using a Verified Protected Network (VPN). Some companies offer this direct-to-consumers to protect their information as well as access location-restricted content. Fraudsters use VPNs to place themselves in low-risk locations to avoid their sender information being flagged as an anomaly. Additionally, they use them to mislead security efforts and mask their location to avoid being traced.

Fraudsters also use geolocation spoofing to place themselves in particular states or countries to take advantage of lessened restrictions in the new geolocation. For example, a user in California spoofed their geolocation to play online poker in New Jersey, taking advantage of New Jersey gambling laws. State law in both states prohibits this, so both states located and apprehended the user. The user forfeited about $90,000 in winnings.

5. DNS Spoofing

Spoofers assume a Domain Name Server (DNS) identity by piggybacking on DNS server caching flaws. As a result, users click on a domain name they trust, but end up on a replica page that leads to phishing or pharming attacks against the user. They click on links within that page and expose themselves to these attacks because they trust the original domain. DNS spoofs, just like many other identity spoofs, often lead to a loss in reputation for the business due to users' trust being violated by the replica site.

This relates to website spoofing, the use of a replica site in order to steal user information. Spoofers target websites that employees use routinely for their work and construct an almost exact replica. Users click on the link to a trusted website, not knowing that the URL is spoofed. They interact with the website, unknowingly entering sensitive credentials or providing backdoor access to their local network. These spoofs are usually most effective when combined with phishing emails.

6. Caller ID Spoofing

Spoofers forge caller ID information, presenting false names or numbers and assuming the identity of particular people or organizations. Public networks and Voice over IP (VoIP) networks make this more possible. Callers answer these, believing their legitimacy, and often share credentials or bank account information due to their trust in the legitimate identity. These calls tend to originate in foreign countries where certain protections may not apply to the caller if they find out that they have been scammed.

7. Email Spoofing

Sender information in the "From" section of an email can be spoofed to hide the origin of fraudulent emails. As long as an email fits the protocols needed by the Simple Mail Transfer Protocol (SMTP) Server, a spoofer easily sends from a falsified email address. The consequences resemble those of IP spoofing and Caller ID spoofing. Spoofers either leverage a man-in-the-middle attack or receive sensitive information, relying on the trustworthiness of the legitimate entity.

8. GPS Spoofing

Although this is a relatively new form of spoofing, it poses an especially dangerous threat. Identity-based GPS spoofing takes the form of a rebroadcast of a genuine signal, or broadcasting fake signals that very closely represent legitimate signals. A spoofer takes on the identity of the trusted GPS satellites, sending falsified or genuine information with malicious intent.

• **MiTm (Man-In-The-Middle) Attack** [12]:

A man in the middle (MITM) attack is a general term for when a perpetrator positions himself in a conversation between a user and an application—either to eavesdrop or to impersonate one of the parties, making it appear as if a normal exchange of information is underway. The goal of an attack is to steal personal information, such as login credentials, account details and credit card numbers. Information obtained during an attack could be used for many purposes, including identity theft, unapproved fund transfers or an illicit password change. Additionally, it can be used to gain a foothold inside a secured perimeter during the infiltration stage of an advanced persistent threat (APT) assault. Broadly speaking, a MITM attack is the equivalent of a mailman opening your bank statement, writing down your account details and then resealing the envelope and delivering it to your door.

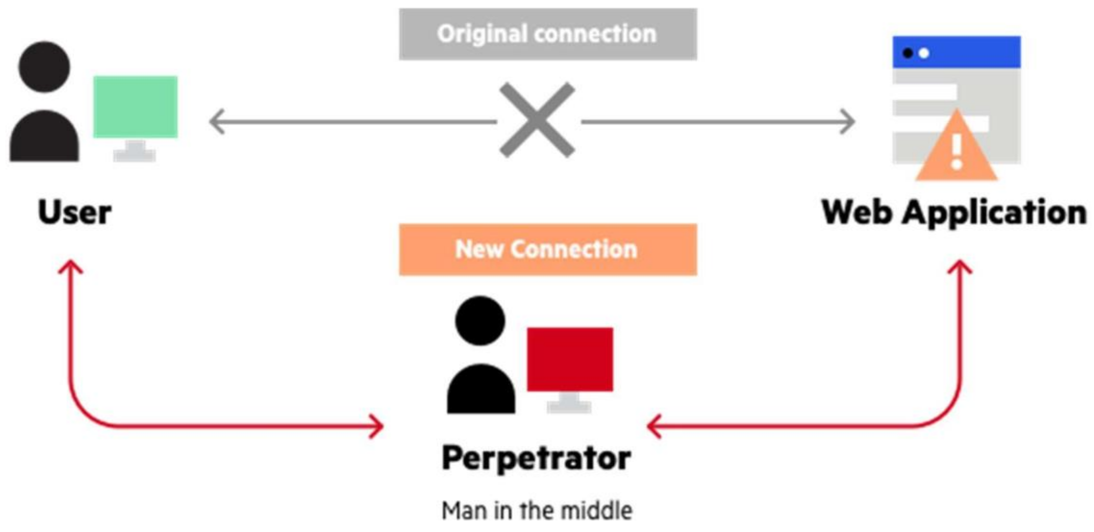


Figure 1: Man-in-the-middle-attack [34]

MITM attack progression

Successful MITM execution has two distinct phases: interception and decryption.

 Interception

The first step intercepts user traffic through the attacker’s network before it reaches its intended destination. The most common (and simplest) way of doing this is a passive attack in which an attacker makes free, malicious Wi-Fi hotspots available to the public.

Typically named in a way that corresponds to their location, they aren’t password protected. Once a victim connects to such a hotspot, the attacker gains full visibility to any online data exchange.

- Decryption

After interception, any two-way SSL traffic needs to be decrypted without alerting the user or application. A number of methods exist to achieve this:

HTTPS spoofing sends a phony certificate to the victim’s browser once the initial connection request to a secure site is made. It holds a digital thumbprint associated with the compromised application, which the browser verifies according to an existing list of trusted sites. The attacker is then able to access any data entered by the victim before it’s passed to the application.

SSL BEAST (browser exploit against SSL/TLS) targets a TLS version 1.0 vulnerability in SSL. Here, the victim’s computer is infected with malicious JavaScript that intercepts encrypted cookies sent by a web application. Then the app’s cipher block chaining (CBC) is compromised so as to decrypt its cookies and authentication tokens.

SSL hijacking occurs when an attacker passes forged authentication keys to both the user and application during a TCP handshake. This sets up what appears to be a secure connection when, in fact, the man in the middle controls the entire session.

SSL stripping downgrades a HTTPS connection to HTTP by intercepting the TLS authentication sent from the application to the user. The attacker sends an unencrypted version of the application’s site to the user while maintaining the secured session with the application. Meanwhile, the user’s entire session is visible to the attacker.

• **Replay Attacks** [13]:

A replay attack is a type of network-based security attack in which the attacker delays, replays, or repeats data transmission between the user and the site. The replay attack in network security is considered lethal because it can resend the message and fool the receiver without decrypting it. Here, the receiver might think that the news is genuine and legitimate.

• **DoS attacks** [14]:

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop.

Popular flood attacks include:

- Buffer overflow attacks, the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks.

- ICMP flood, leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the Smurf Attack or ping of death.

- SYN flood, sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can’t be accessed or used.

- Distributed Denial of Service (DDoS) attack: It occurs when multiple systems orchestrate a synchronized DoS attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once. The distribution of hosts that defines a DDoS provide the attacker multiple advantages:

    - He can leverage the greater volume of machine to execute a seriously disruptive attack.

    - The location of the attack is difficult to detect due to the random distribution of attacking systems (often worldwide).

    - It is more difficult to shut down multiple machines than one.

    - The true attacking party is very difficult to identify, as they are disguised behind many (mostly compromised) systems.

Modern security technologies have developed mechanisms to defend against most forms of DoS attacks, but due to the unique characteristics of DDoS, it is still regarded as an elevated threat and is of higher concern to organizations that fear being targeted by such an attack.

1.3- Cryptography

- Definition

The need for encryption arises when two parties want to communicate with each other via an open channel and still be certain that only they can properly interpret the data and no third party can interfere in any way with the transmitted messages. In addition, it is usually desired for each side to confirm the identity of the other side. There are two main encryption techniques used, symmetric and asymmetric algorithms [15]. Modern cryptography concerns itself with the following four objectives:

- Confidentiality: the information cannot be understood by anyone for whom it was unintended
- Integrity: the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected
- Non-repudiation: the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information
- Authentication: the sender and receiver can confirm each other's identity and the origin/destination of the information.

Procedures and protocols that meet some or all of the above criteria are known as cryptosystems. Cryptosystems are often thought to refer only to mathematical procedures and computer programs; however, they also include the regulation of human behavior, such as choosing hard-to-guess passwords, logging off unused systems, and not discussing sensitive procedures with outsiders.

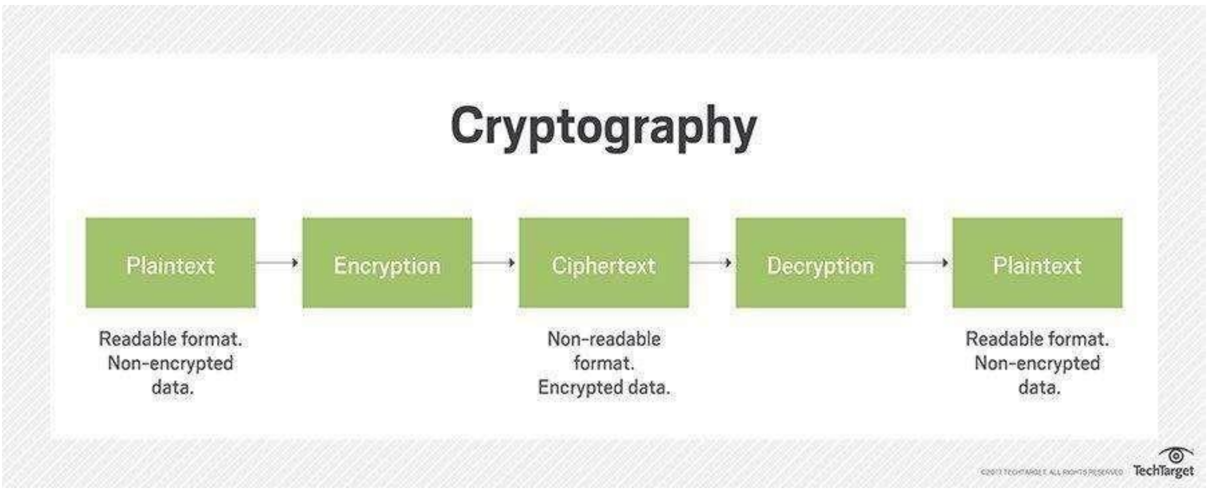

Cryptography
Plaintext
Encryption
Ciphertext
Decryption
Plaintext
Readable format. Non-encrypted data.
Non-readable format. Encrypted data.
Readable format. Non-encrypted data.
TechTarget

Figure 2: Crypthography [35]

- Cryptographic algorithms:

Cryptosystems use a set of procedures known as cryptographic algorithms, or ciphers, to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications. A cipher suite uses one algorithm for encryption, another algorithm for message authentication, and another for key exchange. This process, embedded in protocols and written in software that runs on operating systems and networked computer systems, involves public and private key generation for data encryption/decryption, digital signing and verification for message authentication, and key exchange.

- Types of cryptography:

Cryptography can be broken down into three different types:

- Secret Key Cryptography
- Public Key Cryptography
- Hash Functions

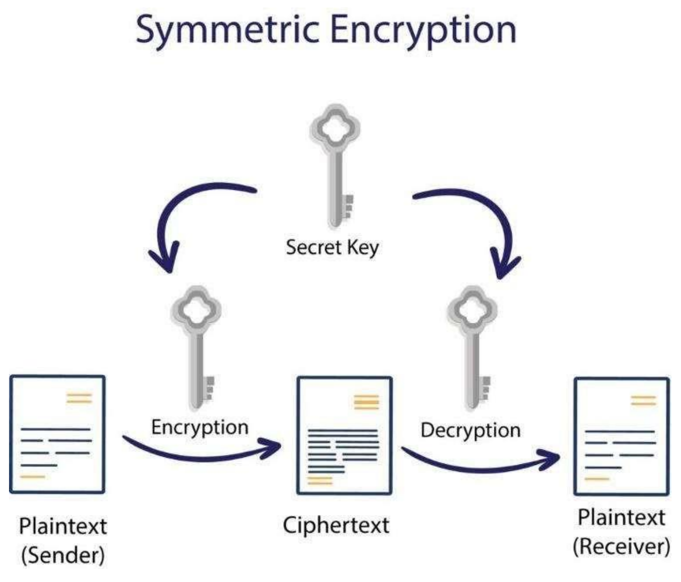


Figure 3: Symmetric encryption [36]

- **Secret Key Cryptography, or symmetric cryptography**, uses a single key to encrypt data. Both encryption and decryption in symmetric cryptography use the same key, making this the easiest form of cryptography. The cryptographic algorithm utilizes the key in a cipher to encrypt the data, and when the data must be accessed again, a person entrusted with the secret key can decrypt the data. Secret Key Cryptography can be used on both in-transit and at-rest data, but is commonly only used on at-rest data, as sending the secret to the recipient of the message can lead to compromise. (Examples AES, DES, Caesar Cipher) [16].

- **Public Key Cryptography, or asymmetric cryptography,** uses two keys to encrypt data. One is used for encryption, while the other key can decrypt the message. Unlike symmetric cryptography, if one key is used to encrypt, that same key cannot decrypt the message, rather the other key shall be used.

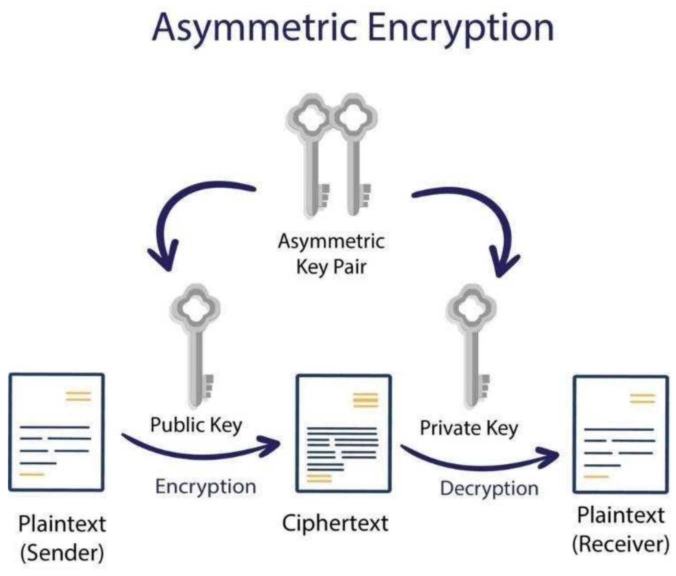


Figure 4: Asymetric encryption [36]

One key is kept private, and is called the “private key”, while the other is shared publicly and can be used by anyone, hence it is known as the “public key”. The mathematical relation of the keys is such that the private key cannot be derived from the public key, but the public key can be derived from the private. The private key should not be distributed and should remain with the owner only. The public key can be given to any other entity. (Examples ECC, Diffie-Hellman and DSS) [17]

- **Hash functions**

Another requirement of secure communication is that the message transmitted is not modified in any way, be it by a malicious third party or by random channel interference/noise.

Cryptography solved this problem by introducing cryptographic one-way hash functions, these functions take as input data and link it with an output of obfuscated text. Given the resulting obfuscated text of a one-way hashing function, it is very difficult to obtain the text used as input to this function. In other words, hash functions provide the fingerprint of the message. A host has to send a packet, as well as its hash, while the recipient has to determine if the packet’s hash matches the one that was sent alongside the packet. If not, then the packet has been modified. Some hashing algorithms are SHA, Whirpool, MD5, HMAC, and MD6 [18].

1.4- Defense mechanisms and practices:

• Introduction

In order to defend against the preceding attacks, computer networks use a plethora of security tools that provide safe communication between two or more parties. Most of these methods use the cryptography algorithms that were described in the previous section. Below are some common security mechanisms that provide defense against attacks in the network:

• **PKI** [19]**:** Public Key Infrastructure can be described as a public dictionary that links public keys with their owners, making it easy for applications to validate the origin of the sender.

• **TLS/SSL** [20]: SSL and its successor TLS are communication protocols that ensure safe transactions via the following steps: (i) initial handshake and selection of common hash and cipher functions; (ii) validation of the server's certificate on the client side (using PKI); (iii) creation and exchange of symmetric key using public key cryptography or Diffie-Helman's algorithm; (iv) communication then initiates while using the symmetric key to encrypt and decrypt data;

• **SSH** [21]: Secure SHell provides secure shell access to a remote host. It achieves that using similar practices as TLS/SSL (without the PKI step).

• **SFTP** [22]: SSH FTP (like FTP), is used for transferring files through the network. However, being built on top of SSH, it also provides reliable data streams between hosts.

• **VPN** [23]: A Virtual Private Network simulates a private network for end users even though data is sent through public or shared networks making sure that unwanted hosts cannot connect or access the shared data. SSH, TLS/SSL, PKI and a variety of other protocols are used to ensure the VPN's functionality.

• **Firewall** [24]: A firewall is the software that controls incoming and outgoing network packets to a system. Through it, the system can block hazardous traffic, untrusted IP's and allow access to sensitive resources only to trustworthy hosts. The firewall consists of a number of rules (also referred to as firewall rules), which are instructions that dictate to the system what should be done with a specific type of packet. The DROP rule, for example, is used in order to deny packets that fit a specific criteria, while the ACCEPT rule acts in the opposite way. Rules are grouped into chains. The three default chains are INPUT (manage incoming packets), FORWARD (manage packets forwarded through a system) and OUTPUT (manage packets sent).

• **Virtual Machine** [25]: A virtual machine is an OS on top of the running OS. It runs on software instead of hardware and if used as a security measure it acts as a 'decontamination chamber'.

• **Reverse-Proxy:** A proxy redirecting requests from clients to a service and vice versa. The clients do not communicate directly with the service and are unaware of its real location and protocols it may uses. Such a mechanism can protect services from attacks and help manage server load.

Some of these mechanisms (such as SSL, AES encryption, Firewall, etc.) have been extensively used throughout this project.

Chapter 2: Authentication prior to connection

2.1- Introduction

In computing, the communications between machines are made via ports, those ports share information of the services that the system provide and also information from the user requesting anything from those services.

To find breaches or exploit vulnerabilities the unauthorized user will start with discovering the provided services that the system offers, scanning open ports is easy, the unauthorized user will send packets to the system open ports and wait for their response which should allow him to find out which service is available on which port, and using this information he can search for vulnerabilities or breaches in the concerned service security allowing him to attack that service.
Tools like Nmap which assist in finding the available ports and services. is applied for locating and abusing vulnerabilities in a machine [5], giving information that will allow the attackers to perform man-in-the-middle attacks, alter transmitted information, sniffing, denial-of-service and gain access using buffing-overflow.

In that case, the system that aim to be accessible must first of all, use firewalls who will protect him from unauthorized users, and give accessibility to only authorized and legitimate users. The difficulty is to identify and authenticate the legitimate users without even establishing connection, in theory we call that “authentication prior to connection “.

So far only two implementations try to achieve that, who are port-knocking (first version) and his improved version Single-Packet-Authorization, which we are using in our project, in the next sections we will explain those two protocols.

2.2- Port-Knocking

• Introduction

Port knocking is a technique suggested as early as February 2003 and has been well documented online by Krzywinski [6,7]. the main purpose of this technique is to secure remote services and hide the information from malicious scans. To achieve that objective all ports should remain closed in the start. In case a user wants to use a hidden service and open the port forwarding traffic to it, a predefined knock sequence must be respected. The port numbers, the order and the time period used in that knock sequence, determine if the client is authorized or not to access the port. If everything matches the predefined sequence, a predetermined port is open, enabling the authorized user to initiate a socket and access the service. This operation is exclusive to the client who did the correct knock sequence, the system will stay closed but also will maintain established connection of the authorized clients, using the knock sequence and the information given once the knock accomplished to authenticate the clients. This technique adds the authentication layer to the system before connection is established.

• Implementation

Programming languages used include C, C++, Perl, Java, BASH, and Python. However, most of implementation are available for Linux/UNIX systems only.

The idea was to use the firewall rules to close all port and incoming connection, the server part software will listen to port knocking operation and try to authenticate a predefined knock based on the order of knocks, their timestamps and the targeted port numbers.

The next step was to create a rule in the firewall allowing exclusively the IP address of the user who did the perfect knock sequence, and send an encrypted UDP telling the user which port is now open and will allow him to access the service.

Those simple operations were able to hide information from attackers and perform the authentication prior to connection, but still the security wasn’t well assured.

The full operation is explained in the figure.

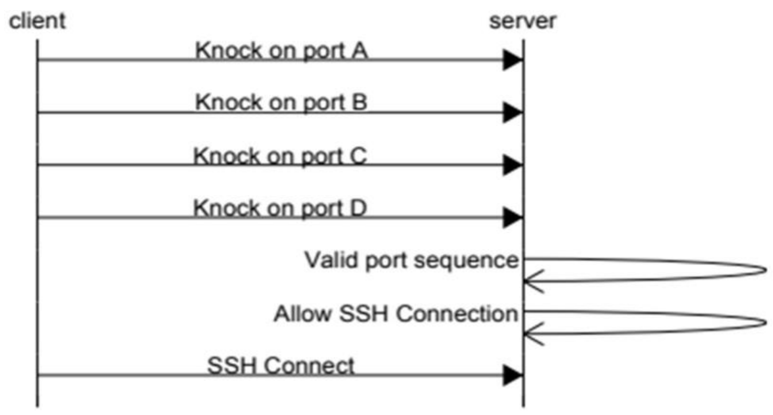


Figure 5: Port Knocking [37]

- **Advantages and disadvantages**

Hiding the information of ports and service behind the firewall and allowing the users to connect at the same time is the main advantage of the port-knocking solution, meanwhile the security level of this solution isn’t enough sophisticated.

Some vulnerabilities was identified in the solution, such as a granted connection by mistake to a user who did a good knock sequence while scanning port only, this vulnerabilities was due to miss configuration in the system side, where the knock sequence was too short, didn’t had a replicate port number inside and the time stamps was disabled. Allowing the port scanning to get access to the service. The most serious one was the easy replay attack to perform and gain access to the service as shown in figure.

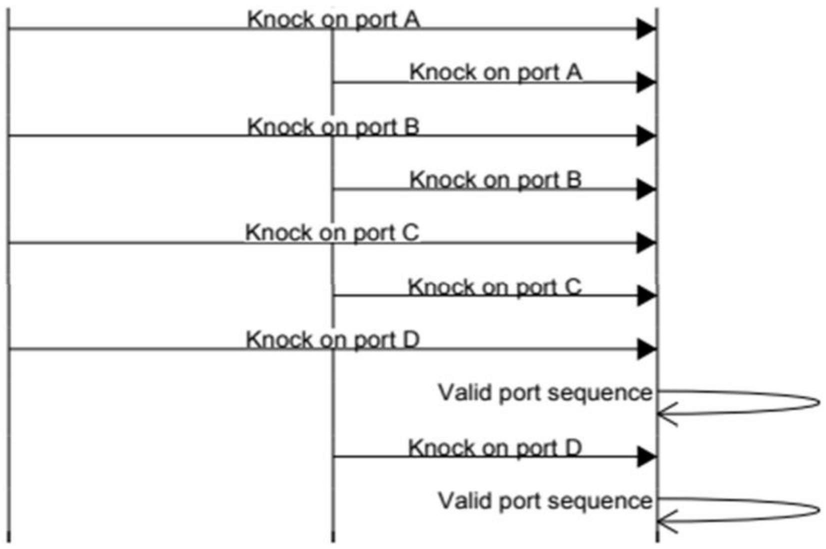


Figure 6: Replay Attack on port knocking [37]

2.3- Single Packet Authorization

• Introduction

The single packet Authorization (SPA) is a solution that allows users to keep services hidden and secured from outsiders, it's the next improved generation of the Port Knocking technique. The goal is to create a strong authentication and authorization layer around the chosen service which transport important information, so this will make the scans and the detection of the concealed port and information impossible even with sophisticated tools.
This is achieved by assuring that the port will be kept hidden until a cryptographically protected key is received to allow the connection to the port, this operation is assured by some commercial solution such as Cryptzone or WebSPA.

In our project we will use the Fwknop implementation, which is open source and available in Github.

• Implementation

The main purpose is to achieve the connection to the server by authenticating the client, doing this is assuming that the communication between client and server is encrypted and decrypted with a shared secret seed.

First of all, the server side will behave just like in port knocking by blocking all port and connections, after that he will be in listening mode, passively he will monitor the network for incoming packets, but instead of searching for port knocks on sequences, he will only look for Spa packets, and once those packets match the requirements, he will add a new rule allowing the connection and enabling the client to connect before closing it again and keep the established connection.
This operation is shown in the figure:

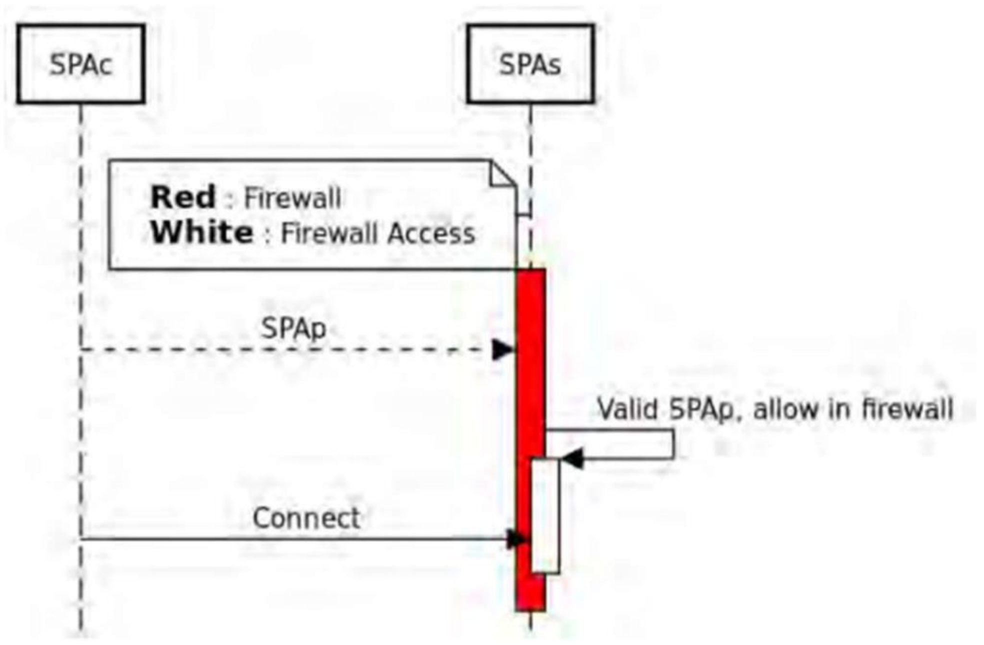


Figure 7: SPA protocol [37]

The client side create his own spa packet to access a specific port on the server, and communicate it with other information to the server, so just like port-knocking the port requested should be known by both client and server, and to do that, the Spa packets must contain those informations:

• AID: the unique ID of the client that sends the packet.

• RANDOM: a random alphanumeric produced by the client.

• PASSWORD: the password of the client (that must be also known by the server).

• NEW_SEED: the new value of the shared seed for the next transaction to simplify connection.

• MD5_HASH: a hash of the previous values to maintain integrity.

To assure the security those informations are encrypted, using the shared secret seed, except for the AID of the client which is sent in plaintext, to make it possible for the server to identify the requesting client and find the right key for decryption.

Once those informations received by the server in the expected format, it will first of all authentify the client using the data encapsulated in the packet, more specifically the AID. Then it will search for the client key, and will use it to decrypt the packets and get the encrypted data, if the decryption is successful the authentication of the client completes

Only when:
- the password in the packet matches the one known by the server for that specific client AID.
- the hash of the received packet matches the MD5_HASH received.
- the random value has not been already used in a previous spa packet sent by the same client while requesting for connection.

After verifying all those information, the connection is made by granting access to the predefined port, and then the server will set the new shared secret key to NEW_SEED value received previously.

- **Changes in new versions:**

In the original SPA protocol, the NEW_SEED value is not included in the SPA packet. It was introduced in this version of SPA in order to provide greater security in the SPA architecture.

SPA packets are exchanged through a public network where any attacker can sniff and capture them. Given that the AID is in plaintext, an attacker that captures the SPA packet knows

The identity of the client that sent it. Then, the attacker can perform a Known Plaintext attack, using as input the SPA packets captured in order to obtain the client’s seed and PASSWORD. This attack, although quite demanding in terms of time and processing resources, is possible, and in fact, it is already widely used against other protocols.

By introducing the NEW_SEED value in the SPA packet, and by letting the server change the seed to NEW_SEED after each successful authentication, Known Plaintext attacks are deemed useless. Even if the secret seed used to encrypt an SPA packet is obtained by the attacker, the next SPA packet sent by the same client will use a different seed for encryption.

- **Conclusion**

The SPA protocol inserts an additional layer of security to a system, and it can be used to protect and secure multiple kinds of services. To achieve this, it makes extensive use of the firewall. The different layers of security created by the SPA are illustrated in the figure.

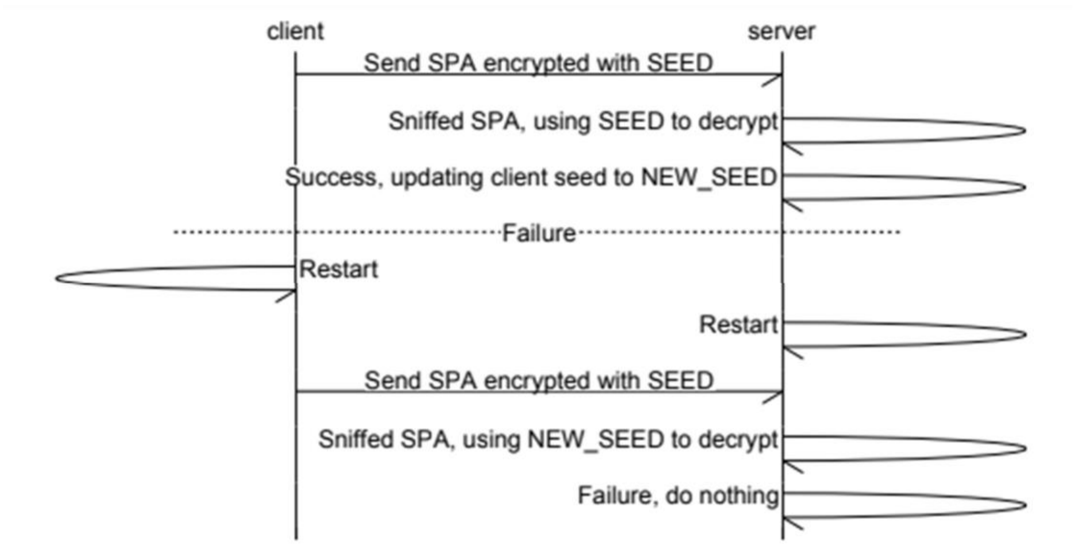

client
server
Send SPA encrypted with SEED
Sniffed SPA, using SEED to decrypt
Success, updating client seed to NEW_SEED
Failure
Restart
Restart
Send SPA encrypted with SEED
Sniffed SPA, using NEW_SEED to decrypt
Failure, do nothing

Figure 8: NEW_SEED illustration [37]

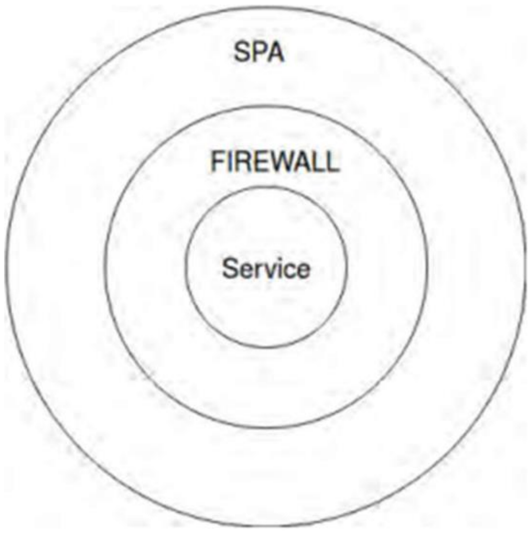


Figure 9: SPA layer of security[37]

- **Spa and port-knocking:**

SPA is a solution that maintains the basic ideas of port-knocking, such as authorization prior to connection and untraceable ports, while trying to address some of its drawbacks.

Some of SPA’s benefits vs. port-knocking are:

• Immunity against Replay-Attacks: By using the RANDOM field, SPA is impervious to Replay attacks.

• Fewer packets: Only one packet sent, compared to the number of packets required by the port-knocking sequence.

• Packet sniffing is harmless: Even if the contents of the SPA packet are acquired through sniffing, the data would be of no use to the attacker since they would be encrypted. However, sniffing port-knocking packets reveals the port-knocking sequence.

• IP spoofing cannot affect authentication: By using IP spoofing, an attacker could disrupt the port-knocking authentication process of a client. However, this does not affect the authentication of clients.

Overall SPA can be considered as an improvement to the port-knocking protocol.

- **Spa limitation:**

Even though SPA seems to provide security against most computer network attacks, high skilled hackers can still exploit the system. An attacker could monitor the network until an SPA authentication attempt is made by a client, wait until the server authenticates the client and allows it in its firewall, and then use IP spoofing to hijack the session. In that case, the problem is solved if the protected service does its own authentication check. However the attacker, instead of trying to hijack the client’s service can also perform a DoS attack on the open port by using IP spoofing. So, the SPA is also susceptible to DoS attacks, but only advanced and skilled hackers can perform them. This issue is also addressed in the next chapter, as SPA cannot solve it using its current structure.

Chapter 3: Software Defined Perimeter.

3.1- Introduction

In recent years there has been a boom in the evolution and proliferation of technology in our daily lives. The number of smart-phones, mobile-connected wireless devices, social networks, and sensors being used has grown substantially due to the emergence of the concept of Internet of Things (IoT).

It was predicted by the National Cable & Telecommunications Association (NCTA) that the number of IoT devices will reach approximately 50 billion by the year 2020 [26].
To handle the increasing number of devices, several technologies and paradigms have been proposed. At the forefront of these technologies and paradigms are cloud computing, software-defined networking (SDN), and network function virtualization (NFV).

Cloud computing has become a main component of the current technology landscape with more than 93 percent of organizations using cloud services in some shape or form [27].
However, the adoption of such technologies and architectures has presented a new set of challenges, in particular those having to do with security. For example, McAfee reported that 52 percent of the respondents surveyed for their report indicated that they tracked a malware infection to a Software-as-a-Service (SaaS) application [27].
Moreover, it was reported that more than 6.1 million DDoS campaigns occurred in 2017 with both the Melbourne IT registrar attack and DreamHost attack being the most prominent [28].

Related challenges include having inadequate trust and authentication models, vulnerability to jamming and sniffing attacks, possibility of data loss or modification, and the possibility of information leakage [29,30,31]. Furthermore, other notable challenges facing such deployments are their vulnerability to man-in-the middle attacks, having inadequate access control measures, and the possibility of network intrusion [32]. As a result, an exploration into new security measures to protect cloud-based networks has commenced, because traditional perimeter defense techniques have proven to be inadequate in protecting the infrastructure from network attacks. A promising solution is a software-defined perimeter (SDP), the concept proposed by the Cloud Security Alliance (CSA) as a security model/framework that has the potential to protect networks in a dynamic manner. This concept was developed based on the Global Information Grid (GIG) Black Core network initiative proposed by the Defense Information Systems Agency (DISA). This model follows a need-to-know model where the device's/application's identity is first verified and authenticated before it is granted access to the application infrastructure. Essentially, the infrastructure becomes "black," meaning, it is undetectable by infrastructures unauthorized to see it. This in turn can help mitigate many network-based attacks such as server scanning, denial of service, password cracking, man-in-the middle attacks, and many others [33].

Next up, the security challenges/concerns facing modern networks are presented.

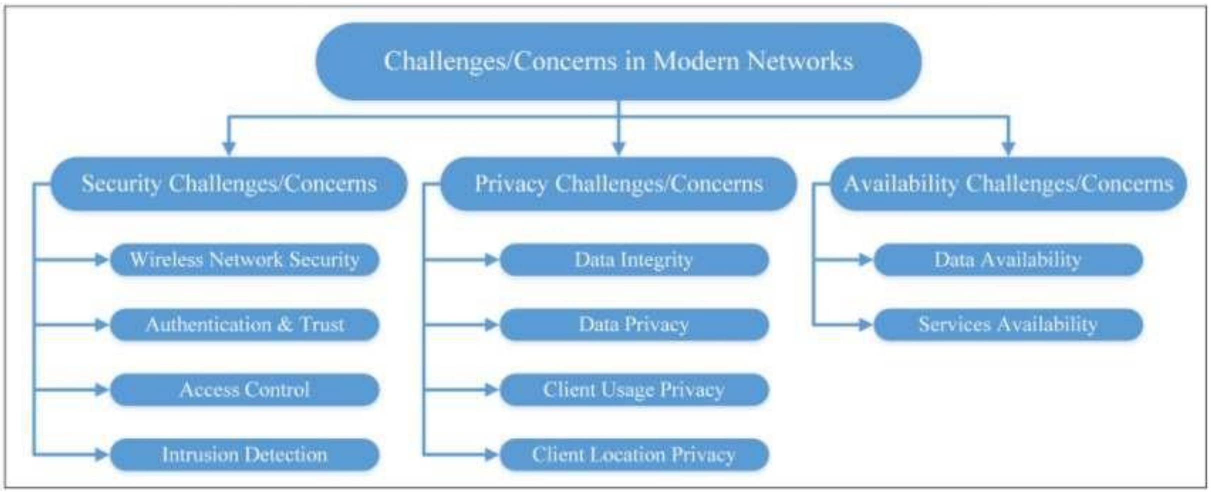


Figure 10: Challenges/Concerns in modern network [38]

Moreover, the SDP framework- as a potential solution is described in more detail in terms of its concept, architecture, and implementations. Also, the anticipated challenges facing the SDP framework are also enumerated. Furthermore, an SDP-based framework adopting client-gateway architecture is proposed. The performance of the framework is evaluated using an internal enterprise scenario (Sonatrach) in terms of connection setup time and network throughput under two types of network attacks, namely denial of service attack and port scanning attack. Therefore, the aim of this project can be summarized as follows:

• Propose the SDP framework as a potential security solution for modern networks in terms of its concept, architecture, and possible implementations.

• Evaluate and analyse the performance of the SDP framework in an internal enterprise scenario using a virtualized network test bed.

As explained earlier in Chapter 1, open network ports in a system can lead to a multitude of separate attacks. SPA was introduced in order to make authentication before connection to a server possible, making the server's ports untraceable by foreign hosts. This concept can be used by various applications as a means to improve security and avoid attacks. The Software Defined Perimeter (SDP), also called Black Cloud, utilizes the SPA protocol in order to provide access to services, by protecting the system from attacks such as server scanning, denial of service, operating system and application vulnerability exploits, and man- in-the-middle attacks (assuming the attacker is not already inside the system).

The Cloud Security Alliance (CSA) has published a software specification document upon which this implementation was based. The CSA's proposal is similar to a DMZ (demilitarized zone) it isolates one or more services behind a perimeter (the SDP) while clients can access and gain access to services only through the gateways of the SDP (as shown on figure).

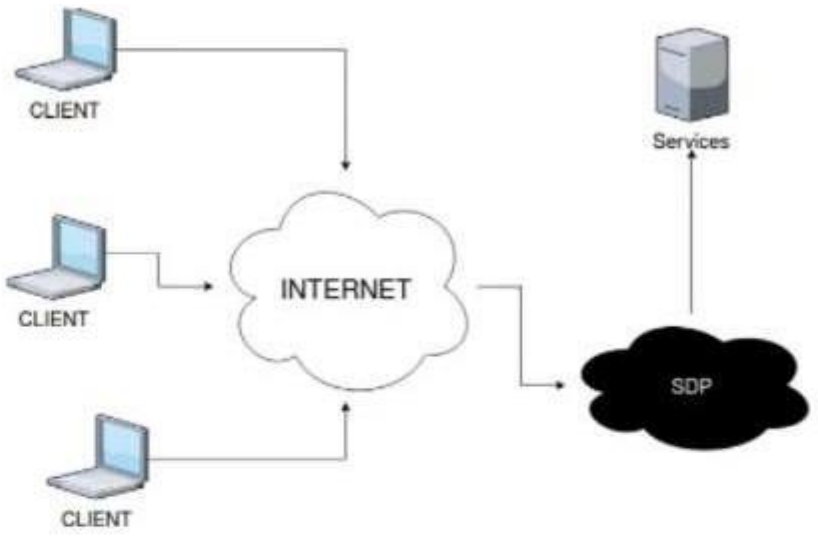


Figure 11: Sdp protection layer [37]

• SDP vs. VPN: What are the differences?

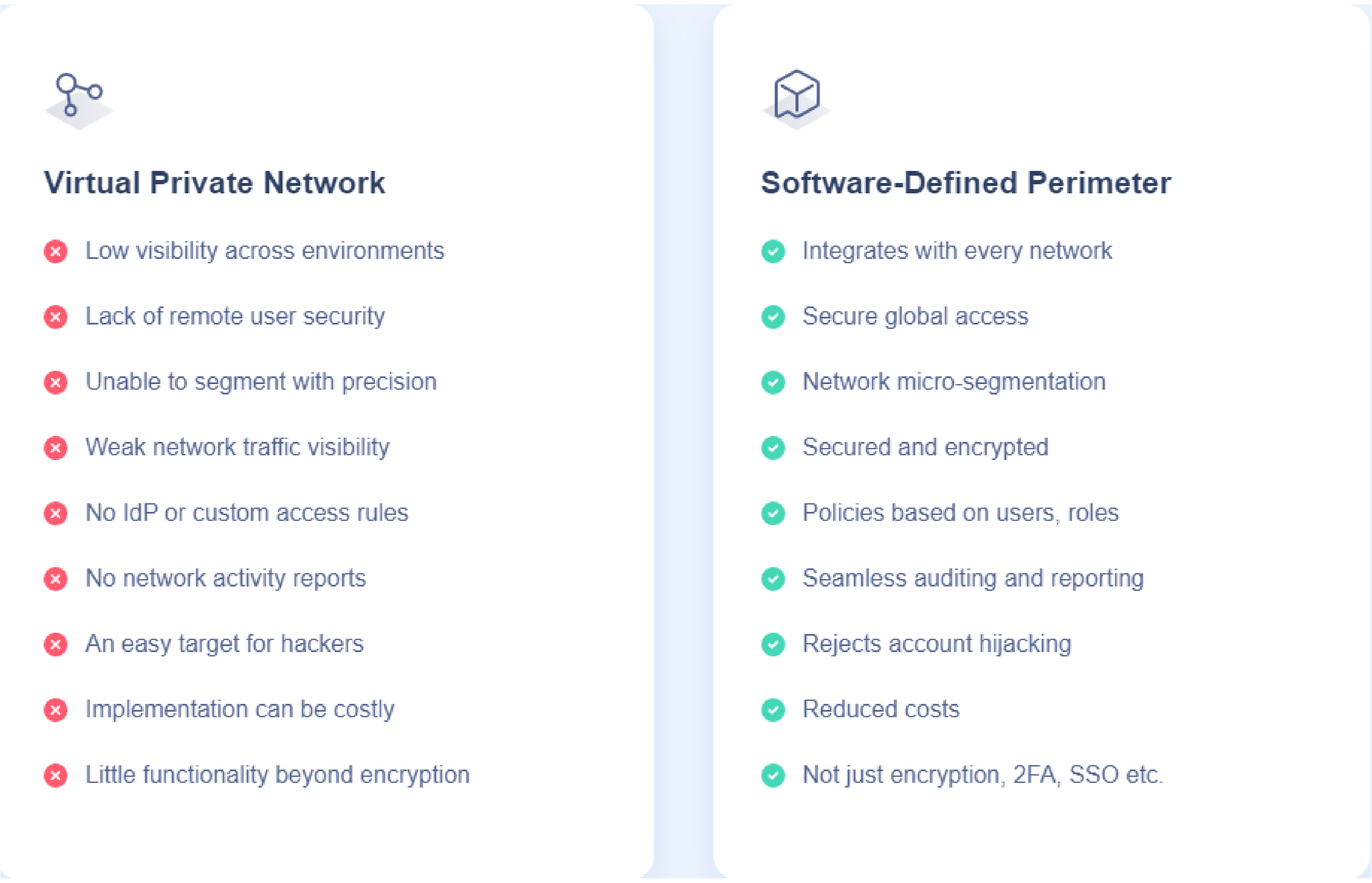
SDPs may incorporate VPNs into their architecture to create secure network connections between user devices and the servers they need to access. However, SDPs are very different from VPNs. In some ways, they are more secure: while VPNs enable all connected users to access the entire network, SDPs do not share network connections. SDPs may also be easier to manage than VPNs, especially if internal users need multiple levels of access.

Managing several different levels of network access using VPNs involves deploying multiple VPNs. Suppose Bob works at Acme Inc. in accounting, Carol works in sales, and Alice works in engineering. Managing their access at the network level involves setting up three VPNs: 1) an accounting VPN to provide access to the accounting database, 2) a sales VPN for the customer database, and 3) an engineering VPN for the codebase. Not only is this difficult for IT to manage, it is also less secure: anyone who gains access to the accounting VPN, for instance, can now access the financial information of Acme Inc. If Bob accidentally gives his login credentials to Carol, then security is compromised — and IT may not even be aware of it.

Now imagine a fourth person: David, the CFO of Acme Inc. David needs access to both the accounting database and the customer database. Should David have to log into two separate VPNs in order to do his work? Or should Acme's IT department set up a new VPN that provides access to both accounting and the customer database? Both options are unwieldy to manage, and the second option introduces a high degree of risk: an attacker who compromises this new VPN with broad access across two databases is able to do twice as much damage as before.

SDPs, on the other hand, are much more granular. There is no overall VPN that everyone who accesses the same resources logs into; instead, a separate network connection is established for each user. It is almost as if everyone has their own private VPN. In addition, SDPs verify devices as well as users, making it much more difficult for an attacker to breach the system with stolen credentials alone.

A couple other key features separate SDPs from VPNs: SDPs are location- and infrastructure-agnostic. Because they are based on software rather than hardware, SDPs can be deployed anywhere to protect on-premises infrastructure, cloud infrastructure, or both. SDPs also easily integrate with multi-cloud and hybrid cloud deployments. And finally, SDPs can connect users in any location; they do not need to be within a company's physical network perimeter. This makes SDPs useful for managing remote teams that do not work within a corporate office.



Virtual Private Network	Software-Defined Perimeter
✗ Low visibility across environments	✓ Integrates with every network
✗ Lack of remote user security	✓ Secure global access
✗ Unable to segment with precision	✓ Network micro-segmentation
✗ Weak network traffic visibility	✓ Secured and encrypted
✗ No IdP or custom access rules	✓ Policies based on users, roles
✗ No network activity reports	✓ Seamless auditing and reporting
✗ An easy target for hackers	✓ Rejects account hijacking
✗ Implementation can be costly	✓ Reduced costs
✗ Little functionality beyond encryption	✓ Not just encryption, 2FA, SSO etc.

3.2- Concept:

The SDP concept is built on the notion of providing application/service owner(s) with the power to deploy perimeter functionality as needed to protect their servers. This is done by adopting logical components in place of any physical appliances. These components are controlled by the application/service owner(s) and serve as a protection mechanism. The SDP architecture only provides access to a client's device after it verifies and authenticates its identity. Such architecture has been adopted by multiple organizations within the Department of Defense in which servers of classified networks are hidden behind an access gateway. The client must first authenticate to this gateway before gaining visibility and access to the server and its applications/services. The aim is to incorporate the logical model adopted in classified networks into the standard workflow (presented in more detail below). Hence, the SDP architecture leverages the benefits of the need-to- know model while simultaneously eliminating the need for a physical access gateway. The general concept is that the client's devices/applications are first authenticated and authorized before creating encrypted connections in real-time to the requested servers. The SDP architecture is composed of and relies on five separate security layers.

- Single Packet Authentication (SPA): SPA is the cornerstone of device authentication. The SDP uses this SPA to reject traffic to it from unauthorized devices. The first packet is cryptographically sent from the client's device to the SDP controller where the device's authorization is verified before giving it access. The SPA is then again sent by the device to the gateway to help it determine the authorized device's traffic and reject all other traffic.

- Mutual Transport Layer Security (mTLS): Transport layer security (TLS) was originally designed to enable device authentication and confidential communication over the Internet. Despite the fact that the standard offers mutual device authentication, it has typically only been used to authenticate servers to clients. However, the SDP utilizes the full power of the TLS standard to enable mutual two-way cryptographic authentication.

- Device Validation (DV): Device validation adds an extra layer of security by ensuring that the cryptographic key used is held by the proper device, because mTLS only proves that the key has not expired nor has it been revoked. However, it cannot prove that it has not been stolen. Therefore, DV verifies that the device belongs to an authorized user and is running trusted software.

- Dynamic Firewalls: In contrast to traditional static firewalls that can have hundreds or thousands of rules, dynamic firewalls have one constant rule which is to deny all connections. The SDP adopts a dynamic firewall policy at the gateway by vigorously adding and removing rules to allow authenticated and authorized users to access the protected applications and services.

- Application Binding (AppB): Application binding refers to the process of forcing authorized applications to use the encrypted TLS tunnels created by the SDP. This is done after the device and the user are properly authenticated and authorized. This ensures that only authorized applications can communicate through the tunnels while unauthorized applications are blocked.

These protocols combined make it extremely challenging for malicious users and attackers to access protected applications and services. Consequently, the SDP framework can address many of the aforementioned security, privacy, and availability challenges, including authentication and trust, access control, data privacy, data availability, and services availability. It is worth mentioning that the complexity of this framework is mainly based on that of the hash function used as part of the encryption/ decryption of the SPA packet since the other four security layers of the framework already exist. For example, dynamic firewalls are used in layer 3. In contrast to traditional static firewalls deployed in existing systems which can have hundreds or thousands of rules, dynamic firewalls have one constant rule which is to deny all connections. The proposed framework adopts a dynamic firewall policy at the gateway by vigorously adding and removing rules to allow authenticated and authorized users to access the protected applications and services. This does not add to or change the existing system’s complexity.

3.3- Architecture:

SDP consists of two components, the SDP Controller (sCTL) and the SDP Hosts.

An SDP host can either be an Initiating Host (IH) or an Accepting Host (AH), the SDP Controller (sCTL) manages and authenticates SDP hosts while it also decides which of them communicate with each other.

IH represents the client entity, it can request access and communicate with services via the SDP system.

The SDP uses AH as gateways to services (thus they are also referred to as SDP Gateways). Their function is similar to a reverse-proxy: they redirect data received from clients to the corresponding services and then forward the services responses back to the clients.

The phrase ’AH protects a service’ is used to symbolize that an AH is configured to act as a gateway for a service.

The AH connects and stays connected to a sCTL until its termination, and the IH connects to the sCTL in order to search for available services.

When the IH selects the service, it wants to use, the sCTL instructs it to connect to the corresponding AH which provides these services, so the full architecture can be seen on Figure.

Both AH and sCTL use the SPA, developed through this project, in order to authenticate their clients. This means that once started, both hosts will configure their firewall with a strict DROP-all-packets rule. Later on, firewall access will be provided only to authenticated clients.

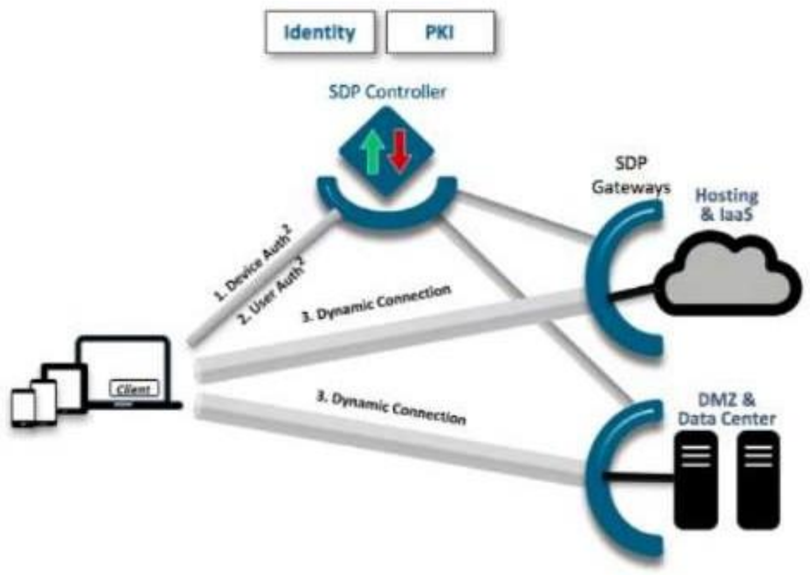


Figure 12: SDP architecture as illustrated on CSA whitepaper [37]

3.3.1- SDP Protocol:

The SDP protocol defines the authentication, communication and structure of the hosts in the SDP system (IH, AH, sCTL). The main components of this protocol are:

a. AH initialization: Where AH connects to the sCTL and acquires information about the services it is going to protect.
b. IH initialization: Where IH connects to the sCTL and acquires information regarding the available services in the SDP system.
c. AH-IH connection: Where IH connects to an AH.
d. Dynamic tunnel mode: Where an IH uses the AH as a gateway to a service.

- **SDP Authentication:**

In the SDP protocol the sCTL serve IH and AH entities, and AH entities serve IH entities. An entity shall be considered as an SDP server if it provides services to other entities, therefore since the sCTL serves IH and AH entities is an SDP server. An SDP client is an entity that is provided services by an SDP server, thus IH is an SDP client. Since AH provides services to IH entities and requests services from the sCTL it is considered both an SDP client (when dealing with the sCTL) and an SDP server (when serving an IH). The SDP Servers use the

SPA protocol as an authentication mechanism, this means that they run an SPA server as a background daemon thread (therefore access to all ports is initially blocked by the SPA server's firewall configuration). Once the SDP client authenticates itself to the SDP Server, they exchange messages over SSL connections. The SDP uses a 2-step authentication mechanism. In the first step the SDP client needs to authenticate to the SPA server, running as a daemon thread on the SDP server, in order to gain firewall access to the SDP Server. On the second step the SDP client needs to authenticate itself to the SDP server as well. This process is illustrated on figure.

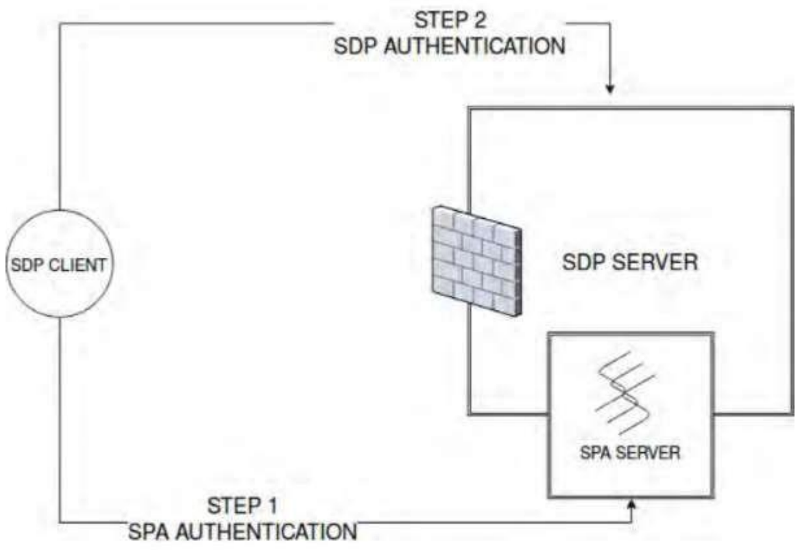


Figure 13: SDP two step authentication [37]

Once the SDP client is authenticated to the SPA server and gains firewall access to the SDP server, it has to authenticate itself once again. In order to authenticate to the SDP server, the SDP client must first connect to the SDP server, through an SSL connection and send a LOGIN or OPEN_CON_REQ packet. The LOGIN packet is used for IH-sCTL and AH-sCTL communication while the OPEN_CON_REQ for IH-AH. Since the SDP uses the SPA protocol, its hosts are also identified by the identification value that the SPA protocol uses, the AID value. For this reason, the LOGIN and the OPEN_CON_REQ packets contain the AID of the SDP client that wants to login to the SDP server and another field called E_AID which is the result of encrypting the client's AID value by using the secret seed that this client shares with the SPA server. The SDP client is considered authenticated to the SDP server if the decryption of the E_AID value produces the AID. If an SDP client is authenticated the SDP server responds with an LOGIN_RESP (response to the LOGIN packet) or an

OPEN_CON_RESP (response to the OPEN_CON_REQ packet). On the other hand, if SDP Client fails to authenticate itself, the SDP Server terminates their connection. The entire process is depicted on figure 14.

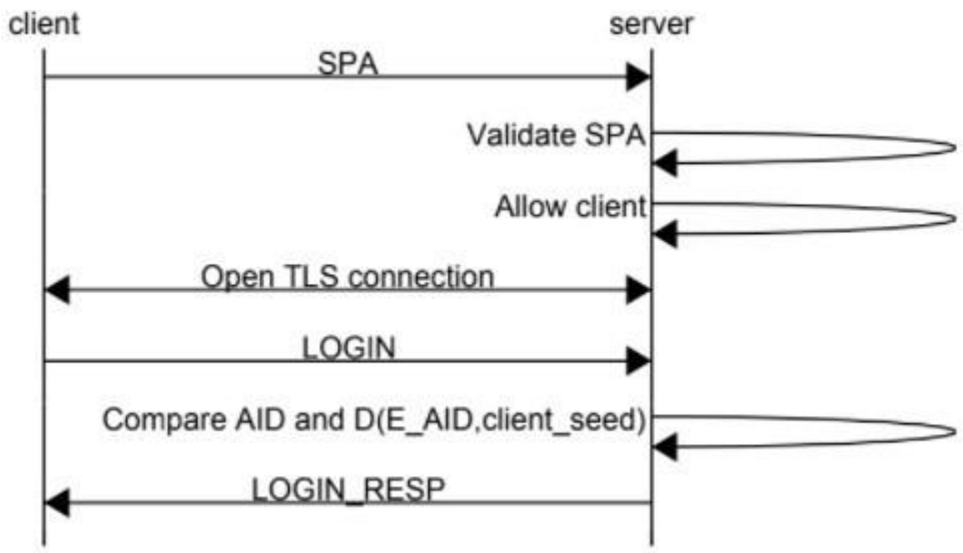


Figure 14: Client authentication to server[37]

- **AH INITIALIZATION:**

Once an AH has been initiated it needs to connect with a sCTL. The first step in this process is to authenticate to the sCTL's SPA server by sending a SPAp. Afterwards the AH sends a LOGIN packet to authenticate to the sCTL and complete the 2-step authentication process. Once the AH has been authenticated, sCTL will respond with a LOGIN_RESP. Upon authentication of the AH, sCTL checks its configuration files and finds out which services must be provided by this AH. Afterwards the sCTL notifies the AH about these services by sending it an AH_SERVICES packet. An AH_SERVICES packet contains a list of services and information about each one of them such as: service type, IP, port, name and the unique ID of the service. After receiving the AH_SERVICES packet, the AH is ready to act as a gateway to each one of the services include in this packet. The process is depicted on figure 15.

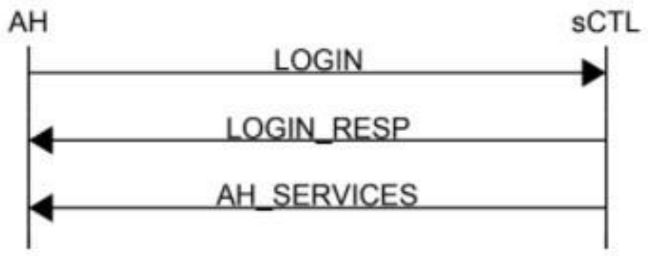


Figure 15: AH Initialization[37]

- **IH INITIALIZATION:**

The IH is the representation of the client of the SDP system. Its first action is to authenticate to the sCTL via the 2-step authentication process (SPA and LOGIN). Afterwards the IH can perform a service query via an IH_QUERY packet. The information included in IH_QUERY packets consist of the fields ’name’, ’type’ and ’service_id’. This packet can be used to find information about available services matching a specific query (by setting the name and type fields of the packet). Once the sCTL receives an IH_QUERY packet, which requests service-specific information, it responds with an IH_SERVICES packet, which contains information regarding the services requested such as their name, type and unique IDs. Another use of the IH_QUERY packet is to explicitly ask for a connection to a specific service by setting the service_id field of the packet to the ID of the corresponding service. Upon receival of such a packet, the sCTL searches for an available AH that is able to provide the service to the IH. It later sends an IH_AUTH packet to this AH to inform it about the IH request to connect with one of the services it protects, this packet contains the AID of the IH, the ID of the requested service and a seed to be used in order to decrypt and encrypt data transmitted during the authentication of this IH. The AH prepares for the connection and notifies the sCTL that it is ready to accept the client with an IHA_ACK. This packet includes the port which will be opened for the IH after it completes its authentication to the SPA server. Finally, the sCTL notifies the IH with an AH_READY packet about the AH that is going to be used as a gateway to the requested service. Information such as the IP and port of the SDP server running on AH as well as the seed to use during its authentication to the AH are included in this packet. When this packet is received by the IH the communication with the AH can finally begin. The work of the sCTL is finished so it terminates the connection with the IH.

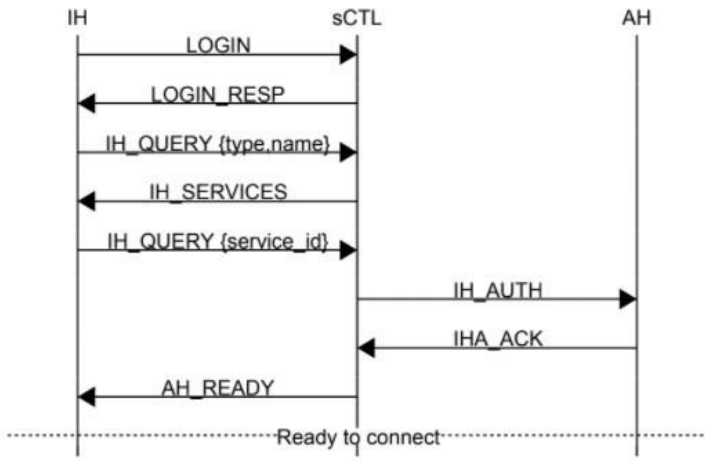


Figure 16: IH Initialization[37]

- **IH-AH Connection/DTM:**

After the IH initialization an AH is ready to accept connections from this IH. However, the IH still needs to perform the 2-step authentication in order to authenticate itself to the AH (in order to prevent session hijacking). Therefore, the IH authenticates itself via SPA to the SPA server background thread of the AH and later on, it sends an OPEN_CON_REQ to authenticate itself to the AH. Throughout the 2-step authentication the IH uses the seed sent to it by the sCTL in the AH_READY packet. Once authenticated the AH will initiate a connection with the service that this IH wants to use. Last and final component of the SDP protocol is the Dynamic Tunnel Mode (DTM). By using the AH as a gateway to the requested service, the IH sends to the service data encapsulated inside SDP packets called DATA packets. The AH will later unpack the SDP packet and send its payload to the service, while also forwarding its response back to the IH. When the IH has finished communicating with the services it sends a CONN_CLOSE packet to the AH to terminate the connection. The entire process can be seen at figure 17.

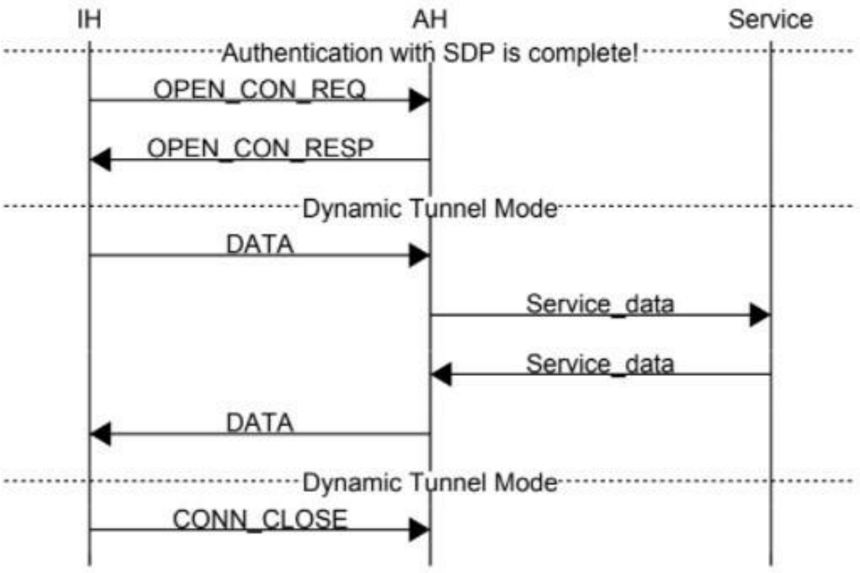
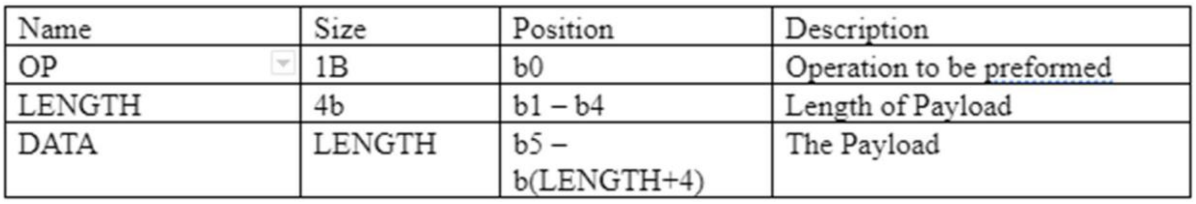


Figure 17: IH-AH Connection and the dynamic tunnel mode[37]

3.3.2- Implementation of the SDP

- **The SDP packet:**

The standard SDP packet is displayed on figure 18 when the packet is received; the receiver handles the data field based on the OP value. The OP value signifies the operation to be performed; available operations are LOGIN, LOGIN_RESP, IH_SERVICES, etc…



Name	Size	Position	Description
OP	1B	b0	Operation to be preformed
LENGTH	4b	b1 – b4	Length of Payload
DATA	LENGTH	b5 – b(LENGTH+4)	The Payload

Figure 18: SDP packet [37]

- **Communication between hosts:**

The hosts use SSL connections in order to communicate with each other. So far only the AH and the sCTL run an SSL server, since they are SDP servers. In this implementation, to save resources and time, all SSL servers (sCTL and connected AHs) use the same pair of private and public keys. In future implementations AH SSL servers could use a different private key which would be obtained by the IH via the AH_READY packet.

- **Acknowledgement Message:**

Even though SSL, provided in the original SDP protocol, offer an efficient and reliable network- level service, an ACK system for the application layer was also required for the new implementation. SDP did not include instructions or details on ACK packages. The figure displays the ACK messages scheme.

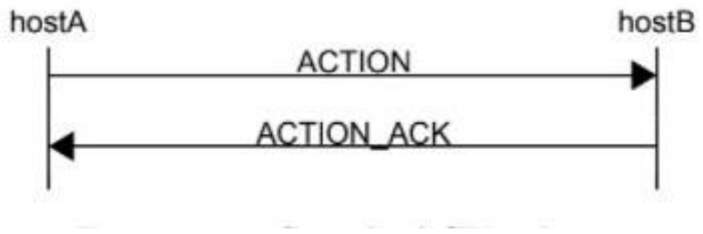


Figure 19: ACK Message [37]

The ACK packets are:

a. AHS_ACK: Sent by the AH to the sCTL to indicate that the list of services has been received.
b. IHA_ACK: Sent by the AH to the sCTL to indicate that it is ready to connect with an IH.
c. LOG_ACK: Sent by an SDP client to an SDP server when the SDP authentication is complete.

• Preventing DOS in the SDP system:

DoS attack is possible on SPA after the SPA server authentication with the SPA client. Since the firewall allows connections from requests originating from the SPA client's IP, an experienced attacker could spoof his/her IP and start a DoS attack on the open port. This is displayed on figure.

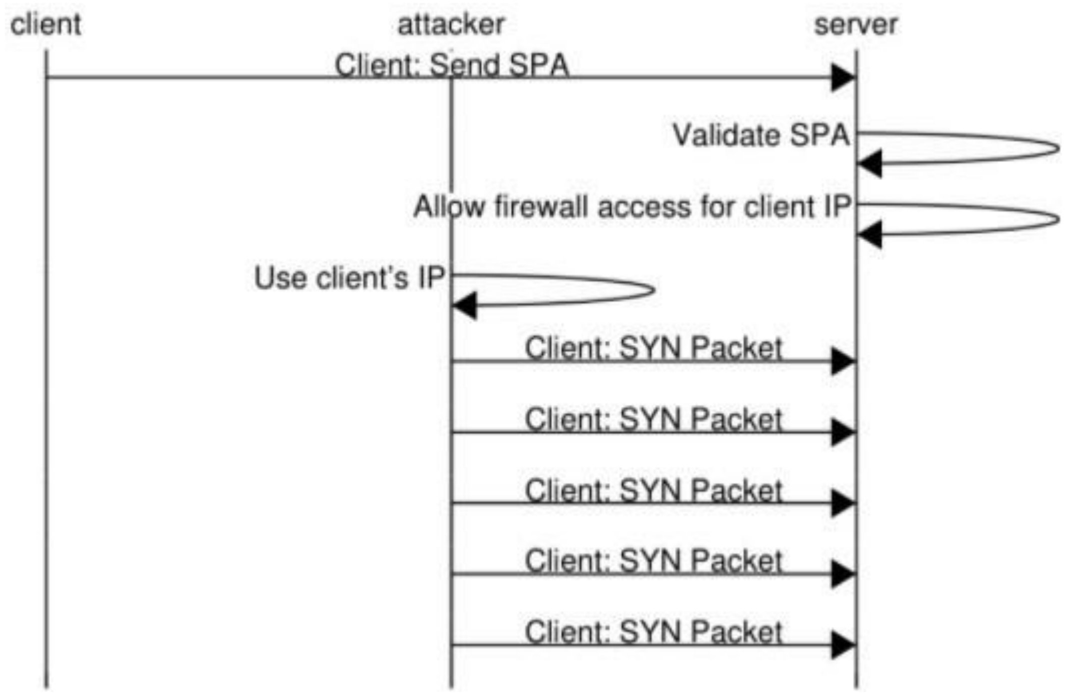


Figure 20: SYN Attack on SPA [37]

Each firewall rule that allows SPA client connections, matches new or established connections. The SDP system stops DoS attacks by setting this rule to match only established connections once the SDP Client has successfully logged in to the SDP Server (by using the set_rule_to_established function of the SPAListener class from spa_lib).

Attackers are not able to authenticate themselves to the SDP system, due to the 2-step authentication mechanism, and during each failed attempt the SDP Server would terminate their connection. Eventually, once the SDP client authenticates itself, the SDP Server will block access to new connections originated by its IP and only the established connection of this client will be preserved. This is displayed on this figure.

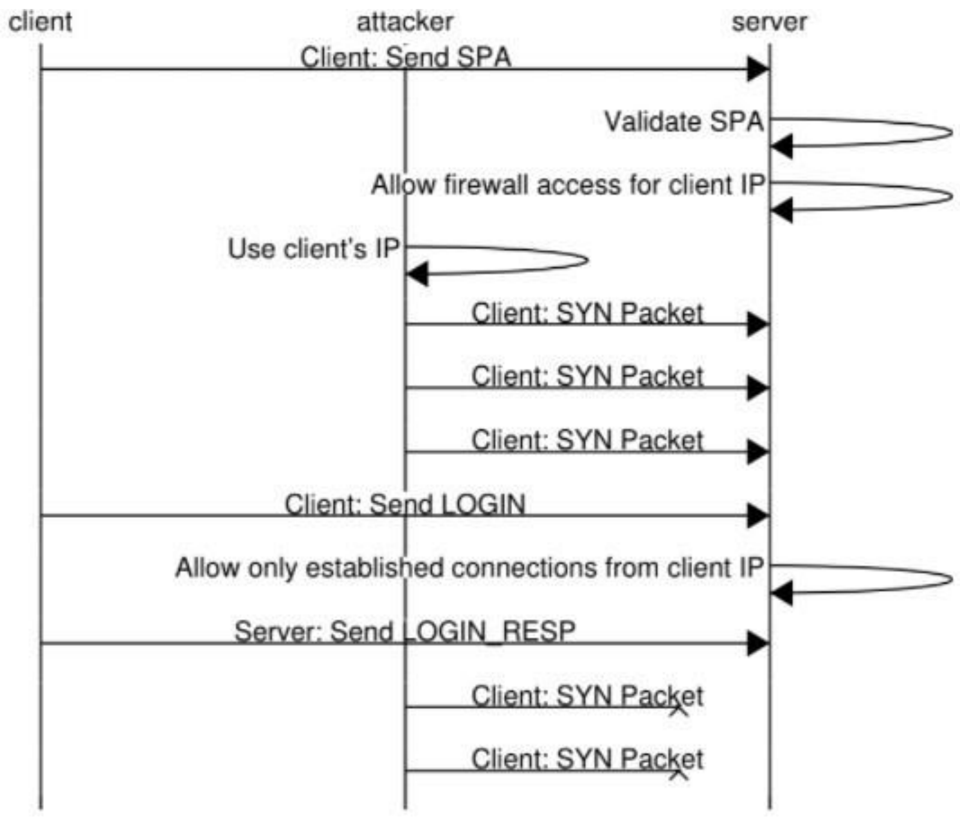


Figure 21: Preventing DOS Attack on sdp [37]

In the scenario where the SDP client does not authenticate itself to the SDP server after the SPA authentication, the firewall will still accept connections from the client's IP and the DoS attack would be possible until the client logs in. In this implementation it was assumed that the SDP client will always try to authenticate itself after the SPA packet so no precautions were taken to solve this problem. However, creating a background daemon thread, which blocks access to the firewall if a connection has not been made after a specific period of time, would easily solve the problem.

3.4- Conclusion

To make this simple, the SDP is composed of three main components: client, gateway and controller. For the purpose of this work, the SDP architecture proposed and implemented is the client-gateway architecture as shown in Fig 22 which can faithfully describe an internal enterprise network scenario. In this case, the gateway also acts as the application server and hosts the service to be accessed. The functions of the three components are as described below:

- **Controller**: The controller is the main component of SDP. It contains the details of the authorized clients and servers, provides the details of rules to the gateway and controls the authentication of each component. The controller proposed in this work makes use of a database for all the above purposes. The database contains the details of all the hosts involved, which is then sent to the gateway. It authenticates these hosts with the help of certificates.
- **Gateway**: The gateway enforces the rules which prevent any unauthorized access to the service hidden behind it. By default, the gateway blocks all traffic. However, once the controller provides the list of authorized initiating (clients) and accepting hosts (servers) and the list of services, it sets up rules which allow a connection to be established between the two while preventing all other traffic. This includes any attempt by the authorized hosts to establish a connection to a service they are not authorized to access. In this work, these rules are set up using the iptables.

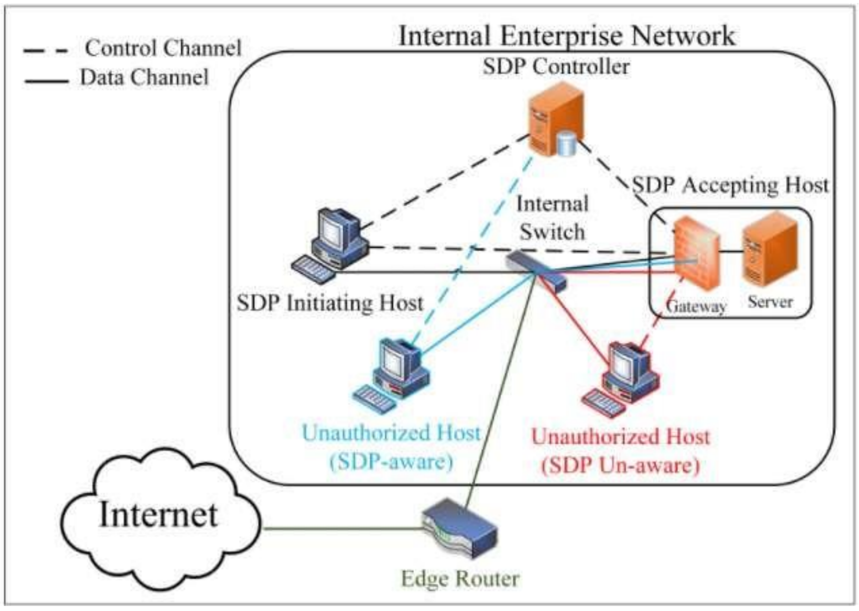


Figure 22: Proposed architecture for internal sdp network [38]

- **Client**: The client is the machine trying to access a service. In SDP, the client first connects to the controller and informs it about the service it wishes to access. Once the verification is complete, it attempts to connect to the service hidden behind the gateway. The gateway will allow the connection request to go through and thus, the client-server data transfer can take place. Ideally, once the connection is established it should not be reset until specifically requested. The whole process of SDP, as shown in Fig. 23, involves the following steps:

- The gateway initiates a TLS connection to the controller and sends an SPA packet.
- The controller verifies the gateway using a certificate present in the gateway.
- The controller establishes a mTLS secured connection between itself and the gateway.
- The controller then proceeds to send all the information about the initiating and accepting hosts as well as the authorized services to the gateway.
- The client initiates a TLS connection to the controller and sends its own SPA packet.
- The controller verifies the client using a certificate present in the client.
- The controller establishes an mTLS secured connection between itself and the client.
- The client sends another encrypted authentication SPA packet to the gateway.
- The packet is decrypted with the keys provided by the controller.
- The information in the decrypted packet is then cross checked with the information it received from the controller.
- The gateway sets up the corresponding firewall rules after a positive crosscheck.
- The client attempts to connect to the service.
- The connection is established once the gateway allows the connection request to pass and data transfer can take place.

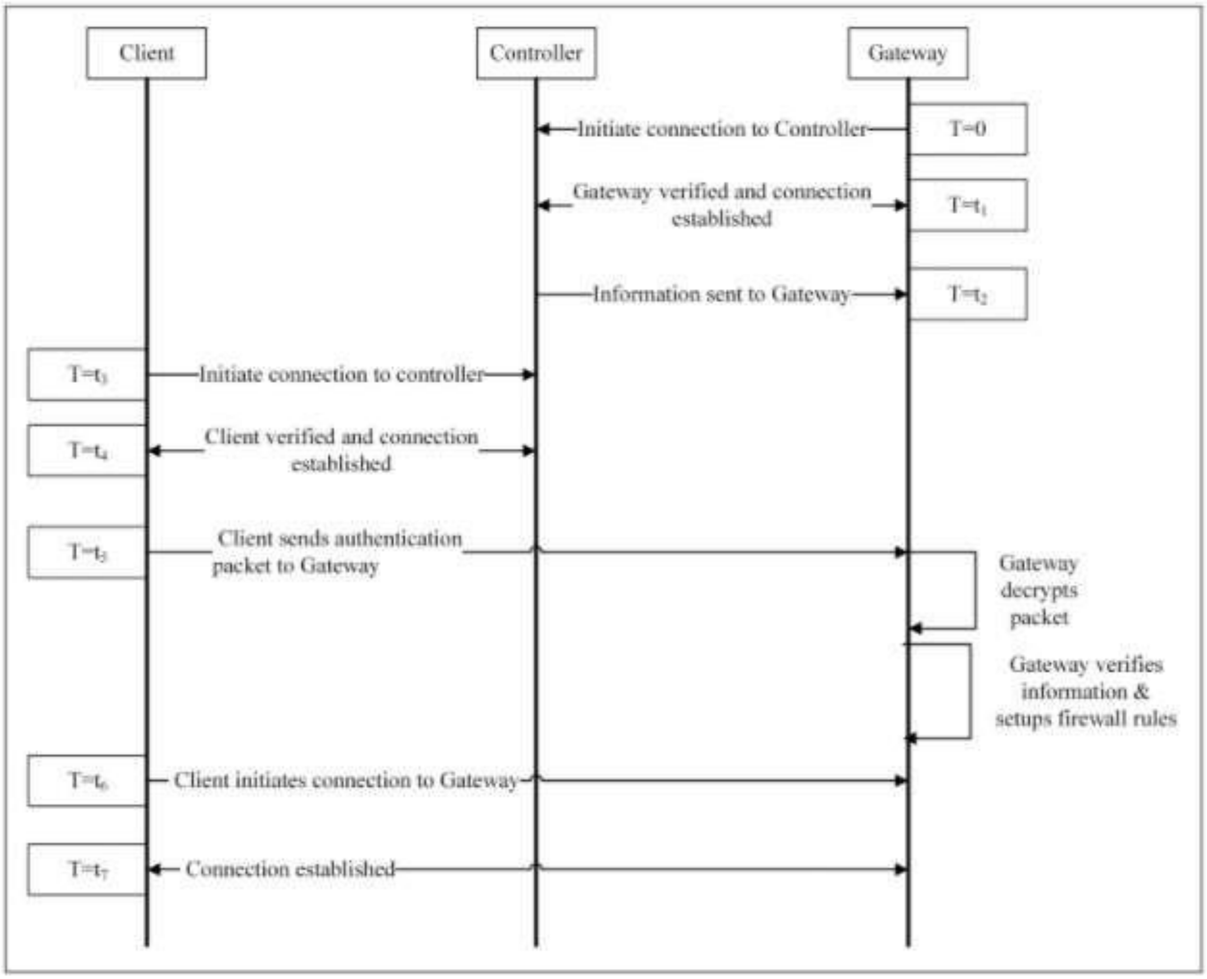

Client
Controller
Gateway
Initiate connection to Controller
T=0
Gateway verified and connection established
T=t1
Information sent to Gateway
T=t2
T=t3
Initiate connection to controller
T=t4
Client verified and connection established
T=t5
Client sends authentication packet to Gateway
Gateway decrypts packet
Gateway verifies information & setups firewall rules
T=t6
Client initiates connection to Gateway
T=t7
Connection established

Figure 23: proposed framework process workflow [38]

Chapter 4: Implementation of Software Defined Perimeter

4.1- Introduction

In this Chapter we will be testing free open-source software published by Waverley Labs, LLC that consists of three modules, a Control module for SDP as the controller, Fwknop module for clients and Fwonopd module for gateways.

This implementation will be tested on a local network using multiple virtual machines.

4.2- Resources:

- **Material Resources**

During the various stages of implementation, training and testing, we used the material resources of our personal station which has the following specifications:

Machine 1

Processor	Intel i5-9400F @ 2.90GHz
Ram	16.0 Go
Operating system	Microsoft Windows 10 x64
Graphic card	NVIDIA GeForce GTX 1650

Machine 2

Processor	Intel i7-8700k @ 3.70GHz
Ram	16.0 Go
Operating system	Microsoft Windows 10 x64
Graphic card	NVIDIA GTX 1080 MHz

- **Software Resources**

For the implementation and testing we used a lot of software resources.

Below are several titles representing the different software used during the development and testing of our application:

- **VMware Workstation Pro** is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems; it enables users to set up virtual machines on a single physical machine and use them simultaneously along with the host machine.

- **Linux Operating System** is an open source operating system. Like other operating system (such as Windows) Linux consists of various software components that manage computer hardware resources and enable you to do tasks such as surfing the web or editing a file in a text editor.

Linux is free and open source software, which means that you can use, copy, study, and change the software in any way. It is distributed with the source code so users can view and modify it.

Many versions of Linux exist. Some of the more popular Linux distributions are Debian, Ubuntu and Fedora.

- **Node.js** an open-source, cross-platform, back-end JavaScript runtime environmentthat runs on the V8 engine and executes JavaScript code outside a web browser.

- **PhpMyAdmin** is a free and open source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting service.

- **OpenSSL** is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. It is also defined as an open-source command line tool that is commonly used to generate private keys, create CSRs, install SSL/TLS certificate, and identify certificate information.

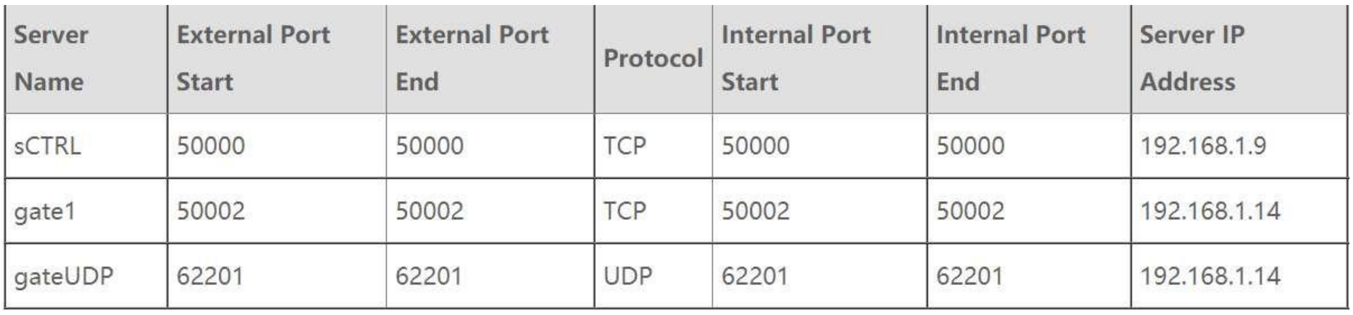
  It is widely used by Internet servers, including the majority of HTTPS websites.

4.3- Installation

In this section we will be discussing the installation process of Controller, Gateway and Client as well as the distribution of the SPA keys and Database content.

- ✓ Each Module is running on a separate Virtual machine with a unique local IP address.
- ✓ 3 ports are needed during these tests.
    - o **Port Forwarding** also called "port mapping," is directing traffic from the outside world to the appropriate server inside a local TCP/IP network, we use this to direct incoming traffic depending on the port used to the desired ip address.

Table 1: Port Forwarding



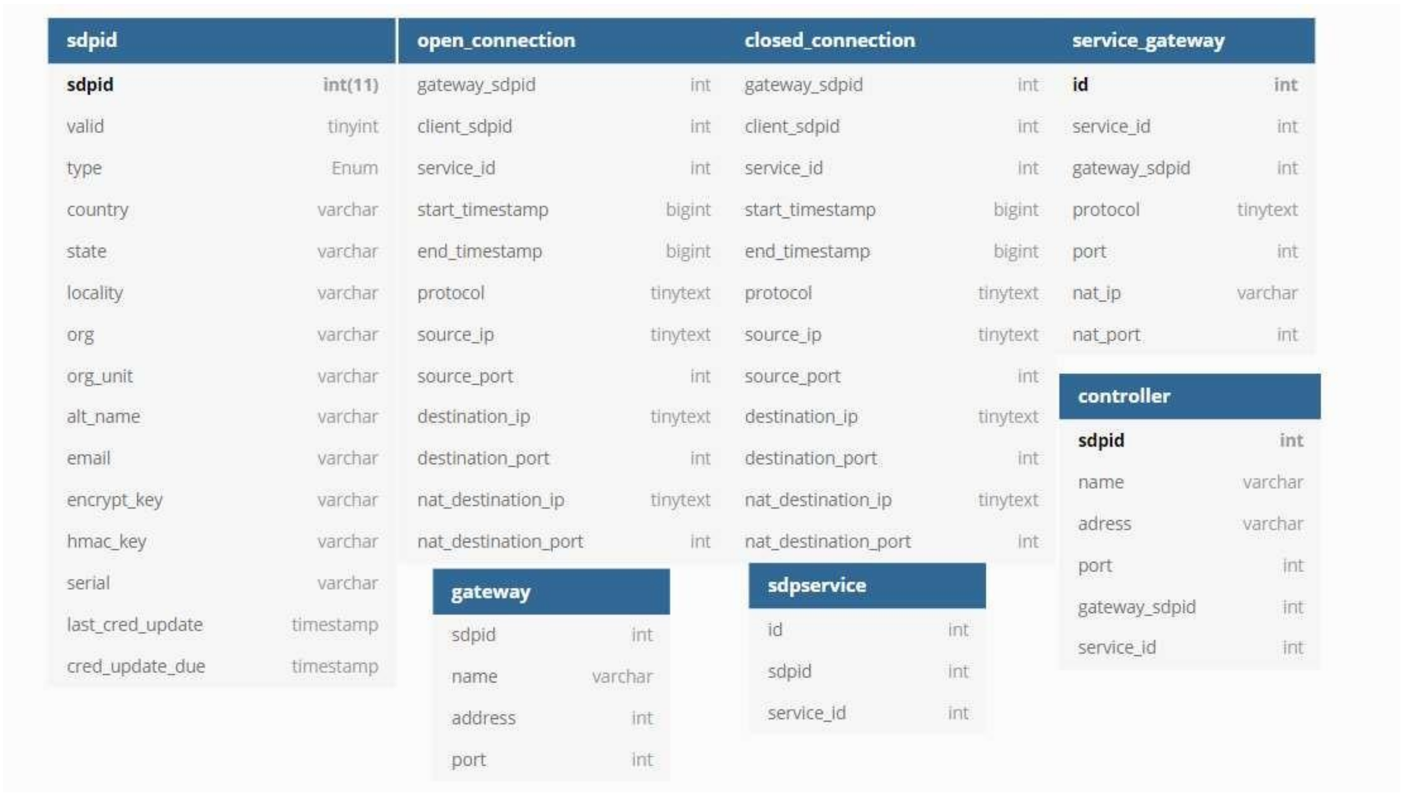
Server Name	External Port Start	External Port End	Protocol	Internal Port Start	Internal Port End	Server IP Address
sCTRL	50000	50000	TCP	50000	50000	192.168.1.9
gate1	50002	50002	TCP	50002	50002	192.168.1.14
gateUDP	62201	62201	UDP	62201	62201	192.168.1.14

A- Controller:

The controller is running on Machine 1 via VMware running a kubuntu-21.04 Linux image hosting a 10.4.20-MariaDB using Apache 8.08-1 distribution.

- Data base :

Table 2: Database tables



sdpid	
sdpid	**int(11)**
valid	tinyint
type	Enum
country	varchar
state	varchar
locality	varchar
org	varchar
org_unit	varchar
alt_name	varchar
email	varchar
encrypt_key	varchar
hmac_key	varchar
serial	varchar
last_cred_update	timestamp
cred_update_due	timestamp

open_connection	
gateway_sdpid	int
client_sdpid	int
service_id	int
start_timestamp	bigint
end_timestamp	bigint
protocol	tinytext
source_ip	tinytext
source_port	int
destination_ip	tinytext
destination_port	int
nat_destination_ip	tinytext
nat_destination_port	int

closed_connection	
gateway_sdpid	int
client_sdpid	int
service_id	int
start_timestamp	bigint
end_timestamp	bigint
protocol	tinytext
source_ip	tinytext
source_port	int
destination_ip	tinytext
destination_port	int
nat_destination_ip	tinytext
nat_destination_port	int

service_gateway	
id	**int**
service_id	int
gateway_sdpid	int
protocol	tinytext
port	int
nat_ip	varchar
nat_port	int

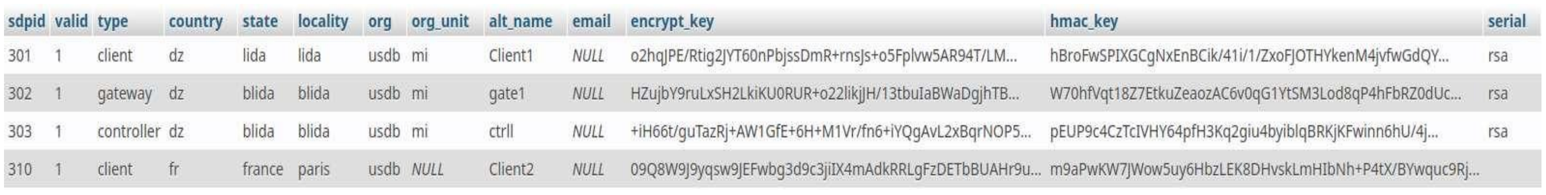
controller	
sdpid	**int**
name	varchar
adress	varchar
port	int
gateway_sdpid	int
service_id	int

gateway	
sdpid	int
name	varchar
address	int
port	int

sdpservice	
id	int
sdpid	int
service_id	int

sdpid – at least one entry for each of the three required components.

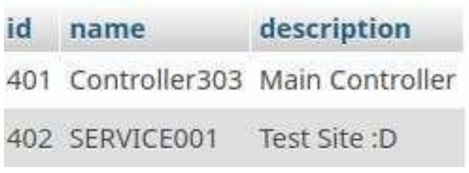
Table 3: Sdpid tables



sdpid	valid	type	country	state	locality	org	org_unit	alt_name	email	encrypt_key	hmac_key	serial
301	1	client	dz	lida	lida	usdb	mi	Client1	*NULL*	o2hqJPE/Rtig2JYT60nPbjssDmR+rnsJs+o5Fplvw5AR94T/LM...	hBroFwSPIXGCgNxEnBCik/41i/1/ZxoFJOTHYkenM4jvfwGdQY...	rsa
302	1	gateway	dz	blida	blida	usdb	mi	gate1	*NULL*	HZujbY9ruLxSH2LkiKU0RUR+o22likjJH/13tbuIaBWaDgjhTB...	W70hfVqt18Z7EtkuZeaozAC6v0qG1YtSM3Lod8qP4hFbRZ0dUc...	rsa
303	1	controller	dz	blida	blida	usdb	mi	ctrll	*NULL*	+iH66t/guTazRj+AW1GfE+6H+M1Vr/fn6+iYQgAvL2xBqrNOP5...	pEUP9c4CzTcIVHY64pfH3Kq2giu4byiblqBRKjKFwinn6hU/4j...	rsa
310	1	client	fr	france	paris	usdb	*NULL*	Client2	*NULL*	09Q8W9J9yqsw9JEFwbg3d9c3jiIX4mAdkRRLgFzDETbBUAHr9u...	m9aPwKW7JWow5uy6HbzLEK8DHvskLmHIbNh+P4tX/BYwquc9Rj...	

service – the controller should be included as an entry in this table.

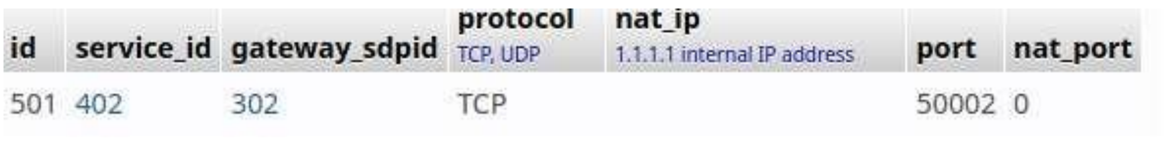
Table 4: Service table



id	name	description
401	Controller303	Main Controller
402	SERVICE001	Test Site :D

service_gateway – for each active service, there should be at least one entry declaring the gateway(s) by which it is protected. There can be multiple instances of a service and/or multiple gateways protecting a service instance.

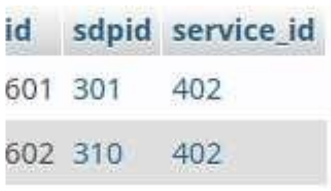
Table 5: Service gateway table



id	service_id	gateway_sdpid	protocol TCP, UDP	nat_ip 1.1.1.1 internal IP address	port	nat_port
501	402	302	TCP		50002	0

sdpid_service – which SDP IDs have access to a service. Remote gateways must have an entry here, providing access to the controller if the controller is behind another gateway.

Table 6: Sdpid_Service Gateway



id	sdpid	service_id
601	301	402
602	310	402

open_connection – will be filled by the controller once all the authentication steps have been met with specific start and end timestamp.

closed_connection – to be filled by the controller once the connection expires and blocks that ip address until the next authentication.

- **Keys Generation**:

OpenSSL version 1.1.1 j 16 Feb 2021 (with 1.1.1c 28 may 2019 Library).

- **RSA**:

RSA algorithm is asymmetric cryptography algorithm.

Asymmetric actually means that it works on two different keys i.e Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private key is kept private.

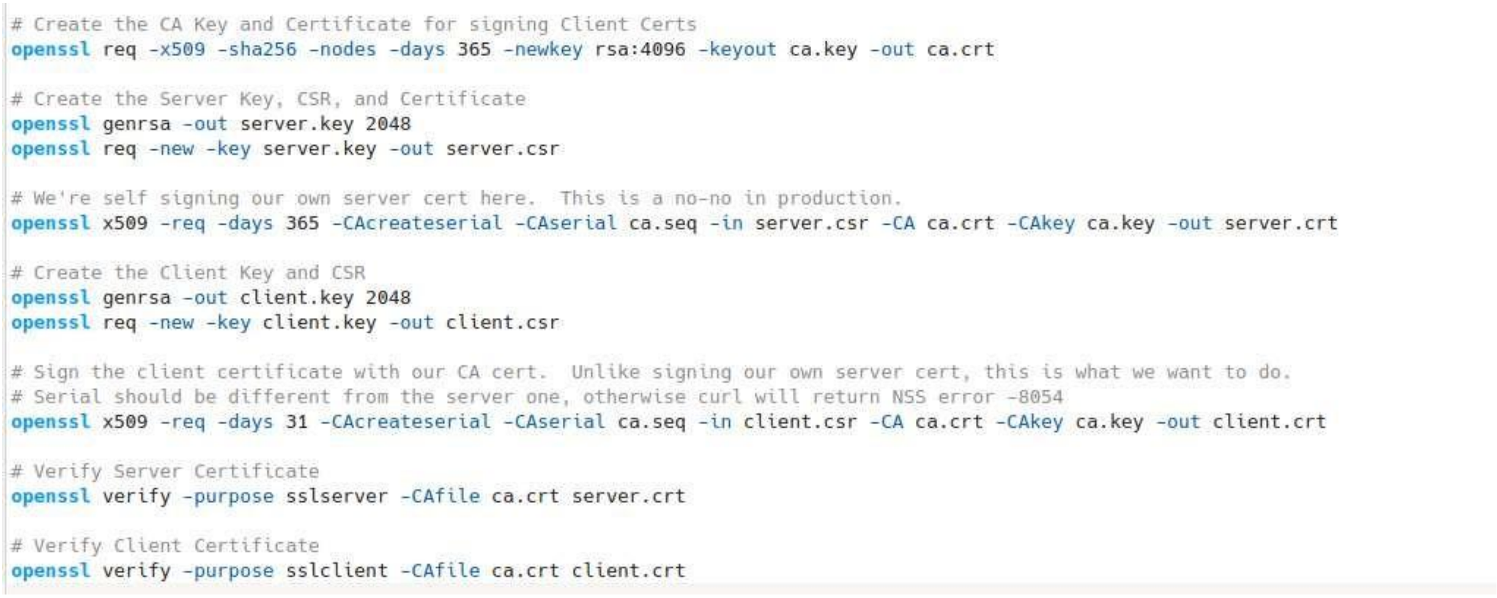
- **Public Keys, Private Keys, and Certificates**:

When performing authentication, SSL uses a technique called public-key cryptography. Public-key cryptography is based on the concept of a key pair, which consists of a public key and a private key.

Data that has been encrypted with a public key can be decrypted only with the corresponding private key. Conversely, data that has been encrypted with a private key can be decrypted only with the corresponding public key.

The controller makes the public key available to anyone, but keeps the private key secret (certificate).

a. A certificate verifies that an entity is the owner of a particular public key.
b. Certificates that follow the X.509 standard contain a data section and a signature section.
c. The controller uses a self-signed certificate rather than a certificate from a Certificate Authority (CA), these certificates are easy to make and do not cost money. However, they do not provide all of the security properties that certificates signed by a CA aim to provide.

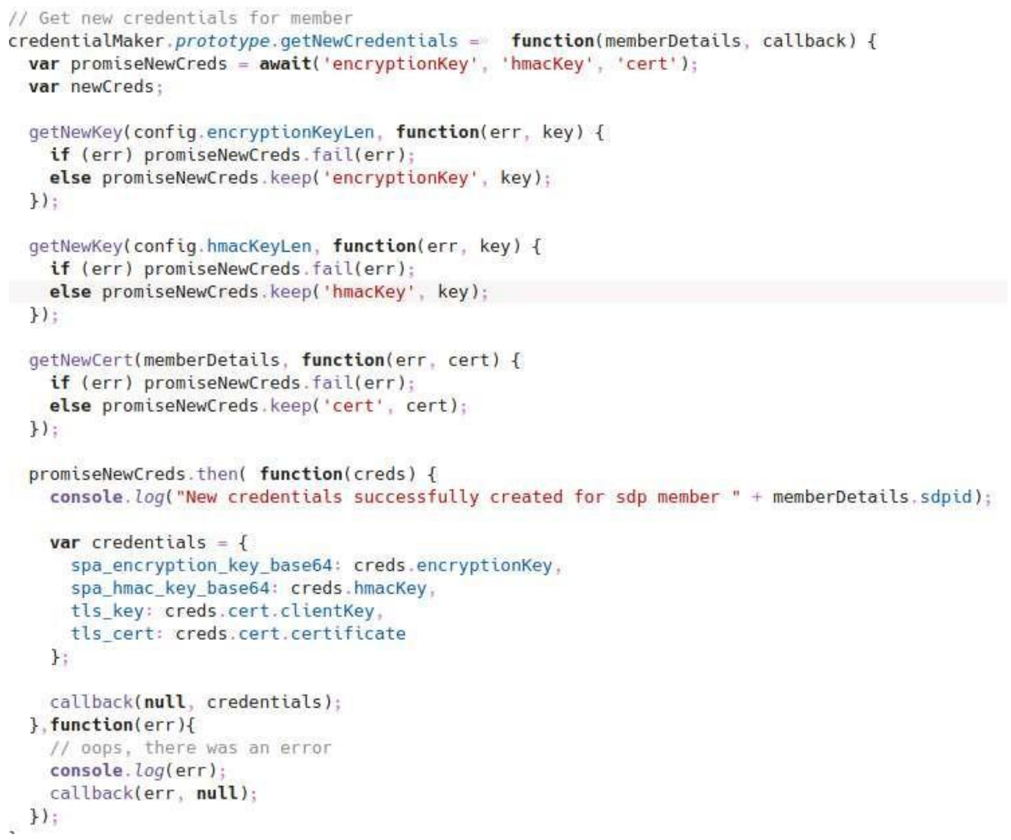


```
# Create the CA Key and Certificate for signing Client Certs
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:4096 -keyout ca.key -out ca.crt

# Create the Server Key, CSR, and Certificate
openssl genrsa -out server.key 2048
openssl req -new -key server.key -out server.csr

# We're self signing our own server cert here.  This is a no-no in production.
openssl x509 -req -days 365 -CAcreateserial -CAserial ca.seq -in server.csr -CA ca.crt -CAkey ca.key -out server.crt

# Create the Client Key and CSR
openssl genrsa -out client.key 2048
openssl req -new -key client.key -out client.csr

# Sign the client certificate with our CA cert.  Unlike signing our own server cert, this is what we want to do.
# Serial should be different from the server one, otherwise curl will return NSS error -8054
openssl x509 -req -days 31 -CAcreateserial -CAserial ca.seq -in client.csr -CA ca.crt -CAkey ca.key -out client.crt

# Verify Server Certificate
openssl verify -purpose sslserver -CAfile ca.crt server.crt

# Verify Client Certificate
openssl verify -purpose sslclient -CAfile ca.crt client.crt
```

Figure 24: OpenSsl spa keys generation

- **Generate sample keys and certs,**

  For each SDP component, after submitting the full set of information about each SDP ID (the controller, gateway, and clients) was entered into the database.

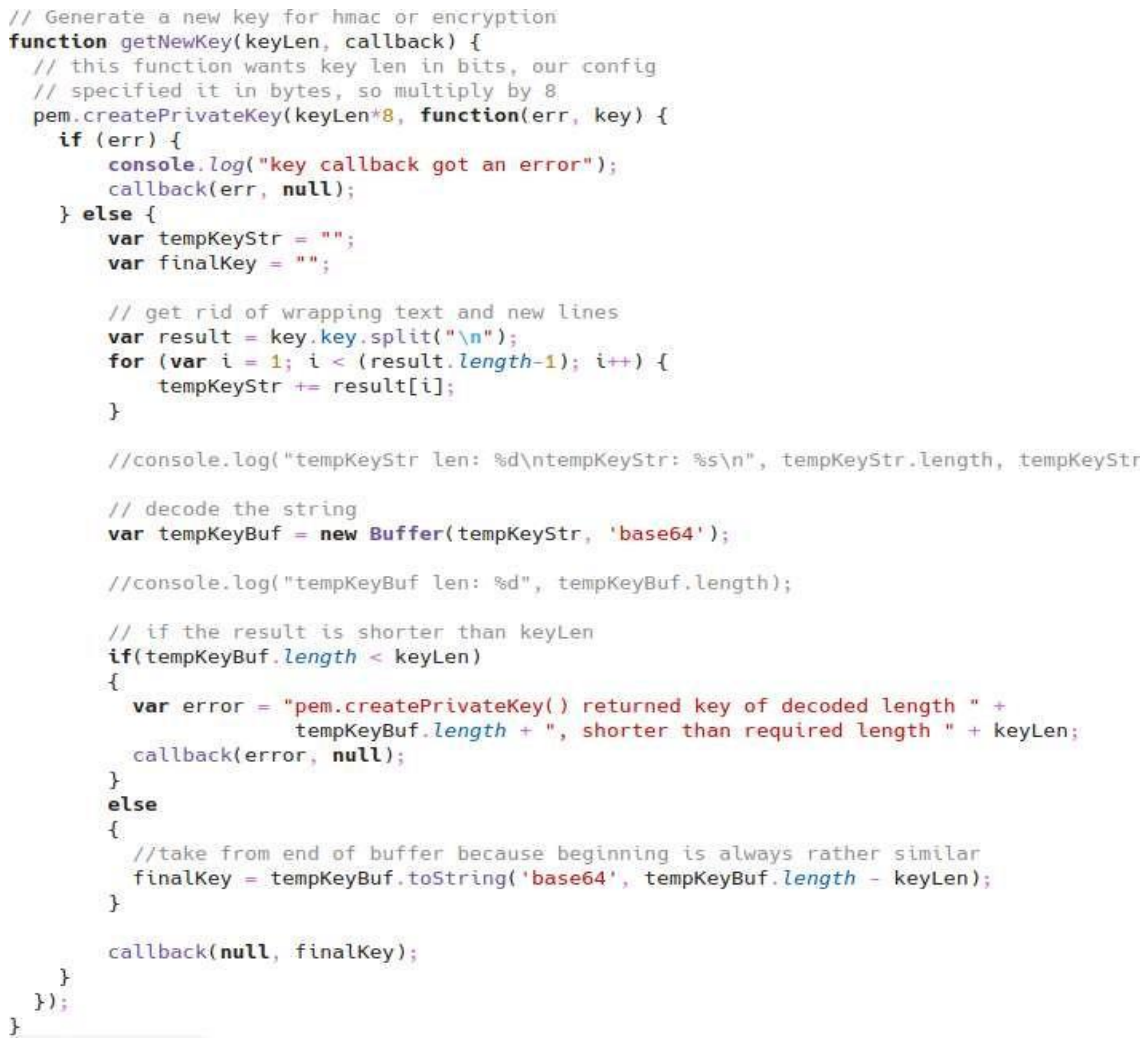


```
// Get new credentials for member
credentialMaker.prototype.getNewCredentials =   function(memberDetails, callback) {
  var promiseNewCreds = await('encryptionKey', 'hmacKey', 'cert');
  var newCreds;

  getNewKey(config.encryptionKeyLen, function(err, key) {
    if (err) promiseNewCreds.fail(err);
    else promiseNewCreds.keep('encryptionKey', key);
  });

  getNewKey(config.hmacKeyLen, function(err, key) {
    if (err) promiseNewCreds.fail(err);
    else promiseNewCreds.keep('hmacKey', key);
  });

  getNewCert(memberDetails, function(err, cert) {
    if (err) promiseNewCreds.fail(err);
    else promiseNewCreds.keep('cert', cert);
  });

  promiseNewCreds.then( function(creds) {
    console.log("New credentials successfully created for sdp member " + memberDetails.sdpid);

    var credentials = {
      spa_encryption_key_base64: creds.encryptionKey,
      spa_hmac_key_base64: creds.hmacKey,
      tls_key: creds.cert.clientKey,
      tls_cert: creds.cert.certificate
    };

    callback(null, credentials);
  },function(err){
    // oops, there was an error
    console.log(err);
    callback(err, null);
  });
```

Figure 25: SDPCredentialMaker.js

The credentials are made using two main functions:

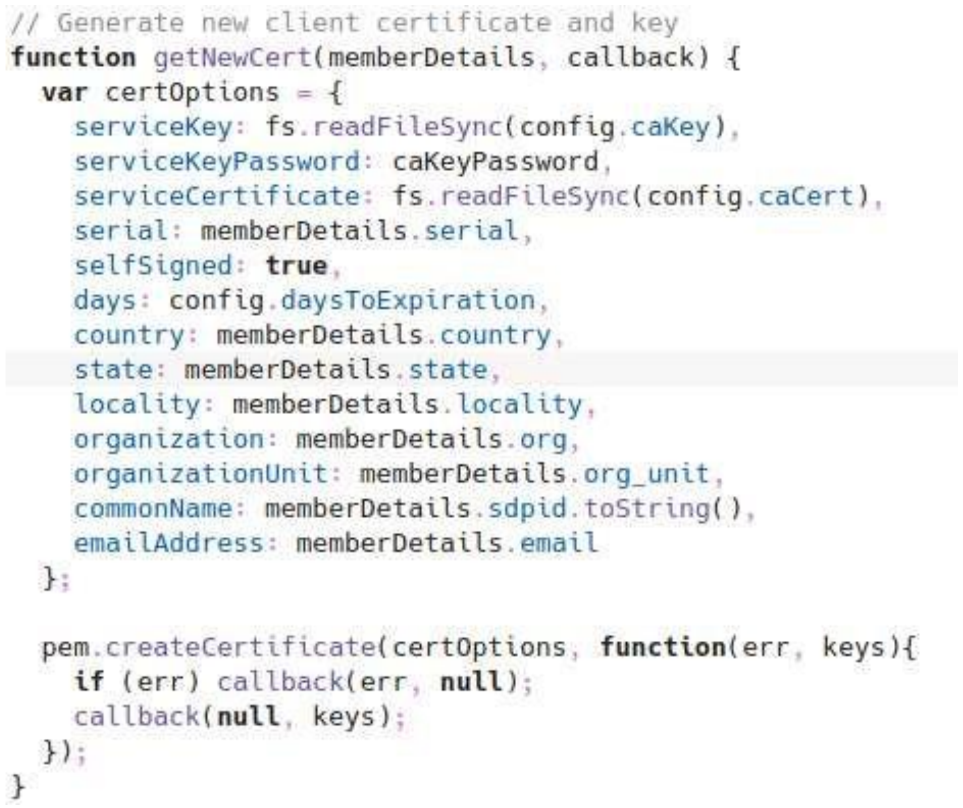
- getNewKey (); to generate spa keys for each module



```
// Generate a new key for hmac or encryption
function getNewKey(keyLen, callback) {
  // this function wants key len in bits, our config
  // specified it in bytes, so multiply by 8
  pem.createPrivateKey(keyLen*8, function(err, key) {
    if (err) {
        console.log("key callback got an error");
        callback(err, null);
    } else {
        var tempKeyStr = "";
        var finalKey = "";

        // get rid of wrapping text and new lines
        var result = key.key.split("\n");
        for (var i = 1; i < (result.length-1); i++) {
            tempKeyStr += result[i];
        }

        //console.log("tempKeyStr len: %d\ntempKeyStr: %s\n", tempKeyStr.length, tempKeyStr

        // decode the string
        var tempKeyBuf = new Buffer(tempKeyStr, 'base64');

        //console.log("tempKeyBuf len: %d", tempKeyBuf.length);

        // if the result is shorter than keyLen
        if(tempKeyBuf.length < keyLen)
        {
          var error = "pem.createPrivateKey() returned key of decoded length " +
                      tempKeyBuf.length + ", shorter than required length " + keyLen;
          callback(error, null);
        }
        else
        {
          //take from end of buffer because beginning is always rather similar
          finalKey = tempKeyBuf.toString('base64', tempKeyBuf.length - keyLen);
        }

        callback(null, finalKey);
    }
  });
}
```

Figure 26: Function that generate keys

- getNewCert (); to generate a certificate for each module



```
// Generate new client certificate and key
function getNewCert(memberDetails, callback) {
  var certOptions = {
    serviceKey: fs.readFileSync(config.caKey),
    serviceKeyPassword: caKeyPassword,
    serviceCertificate: fs.readFileSync(config.caCert),
    serial: memberDetails.serial,
    selfSigned: true,
    days: config.daysToExpiration,
    country: memberDetails.country,
    state: memberDetails.state,
    locality: memberDetails.locality,
    organization: memberDetails.org,
    organizationUnit: memberDetails.org_unit,
    commonName: memberDetails.sdpid.toString(),
    emailAddress: memberDetails.email
  };

  pem.createCertificate(certOptions, function(err, keys){
    if (err) callback(err, null);
    callback(null, keys);
  });
}
```

Figure 27: Function that generate certificates

This code will create three new files sdpid.crt, sdpid.key, and sdpid.spa_keys.It will store the SPA keys in the controller database automatically. The crt/key files and spa_keys must be transferred to the matching clients and gateway’s sdpid.

The main Controller code will be reviewed once the gateway and client modules have been introduced.

Gateway

The gateway is running on Machine 1 via VMware running a kubuntu-21.04 Linux image

Hosting a test web server using:

- Apache2: apache2 is open-source HTTP server. It is still the most popular web-server used worldwide today.
- Php and php sqlite component: PHP is a server-side scripting language. PHP and its component will be used to interact with a backend mySQL database for the test website.
- mySQL: mySQL is a database solution in which we will be storing our data in the table

The gateway will be concealed under Fwknopd server version using a drop all policy Iptables v1.8.7 (nf_tables).

Fwknopd is the server component for the FireWall Knock Operator, and is responsible for monitoring and processing Single Packet Authorization (SPA) packets that are generated by fwknop clients, modifying a firewall ACL (Access Control List) to allow the desired access after authenticating and decrypting a valid SPA packet (in that order), and removing access after a configurable timeout.

The main application of this program is to conceal services such as our test server with an additional layer of security in order to make the exploitation of vulnerabilities much more difficult. In addition, services that are concealed in this fashion naturally cannot be scanned for with Nmap or Shodan.

The main configuration for fwknopd is maintained within 3 files:

- **fwknopd.conf** –which holds all the fwknopd configuration such as the sniffed interface (ens33) and the path to the iptables used, as well as the path to sdp_ctrl_client_conf.
- **gate_sdp_ctrl_client.conf** –contains the following information for connecting to a local or remote controller:
    - CTRL_PORT: Port that SDP Controller is listening on
    - CTRL_ADDR: Address at which the SDP Controller resides

- KEY_FILE, CERT_FILE: Location of the client's own key file and certificate for encrypted communications with the controller.
- CA_CERT_FILE: Location of CA certificate file for verifying peer certificates (the controller certificate)
- And as the Controllers spa keys.

- **gate.fwknoprc**

Only required on remote gateways that must communicate with the controller through the controller's local gateway. Remote gateways must initially send SPA like any other client so that the controller's local gateway opens a path for the remote gateway.

The crt/key files and spa_keys for the gateway will be stored inside the Fwknopd installation file.

Client

The clients are running on Machine 1 and 2 via VMware running an Ubuntu 20.04 LTS (Focal Fossa) Linux image

The client will use Fwknop client version for the authentication and Spa exchange.

The main configuration for fwknop is maintained within 2 files:

- **.fwknoprc**

This file is broken down into "stanzas." Each stanza has a name which is enclosed in brackets. Typically, when calling the fwknop client, the user passes a stanza name to the client. Each stanza tells the client how to generate and send SPA to gain access to a particular.

Stanza architecture:

- ALLOW_IP: The ip address of the client to permit through the firewall.
- SPA_SERVER: The address of the gateway protecting the controller or other service.
- SERVICE_IDS: which service the clients request access to, access will be given depends on the DateBase permissions.
- KEY_BASE64: the spa key of the gateway that the service is concealed under
- HMAC_KEY_BASE64 : the hmac key of the gateway

- SDP_CTRL_CLIENT_CONF: whenever a service has this variable set, by default, the SDP client program will first send a SPA packet to the intended service gateway to enable access to the desired service. The program will then proceed to use the configuration file indicated by this variable to connect to the SDP controller, first using the [sdp_ctrl_gate] stanza to send SPA to the controller gateway and then other settings in the configuration file to complete the connection

- **Sdp_ctrl_client.conf** –This file contains the following controller configuration**:**

  - CTRL_PORT : Port that SDP Controller is listening on
  - CTRL_ADDR: Address at which the SDP Controller resides
  - KEY_FILE, CERT_FILE: Location of the client's own key file and certificate for encrypted communications with the controller.
  - CA_CERT_FILE: Location of CA certificate file for verifying peer certificates (the controller certificate)
  - And as the Controllers spa keys.

4.4- Tests and practices

In order to demonstrate the SDP implementation's usability and features, only two machines were used.

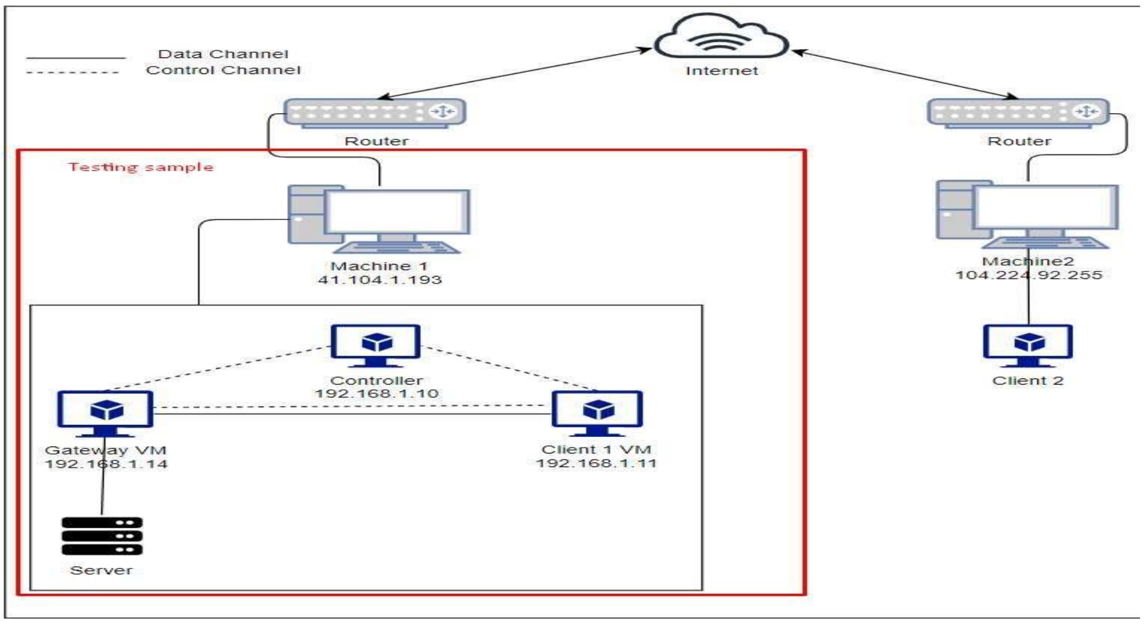


Figure 28: Tests architecture

- **Testing:**

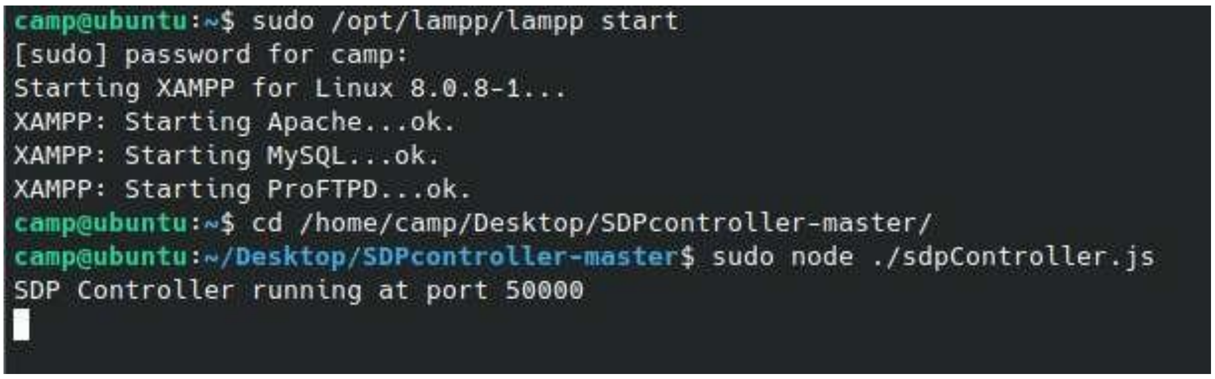
Using a local environment where all 3 modules are running on bridged network where they are connected directly to the physical network and managed by the router NAT (network address translation).

We will be showing a real time results and perspective of each module after each command and action taken.

➢ **Initiation**

A- Controller

Turning on the Data Base and running the main controller loop which is running on port 50000.

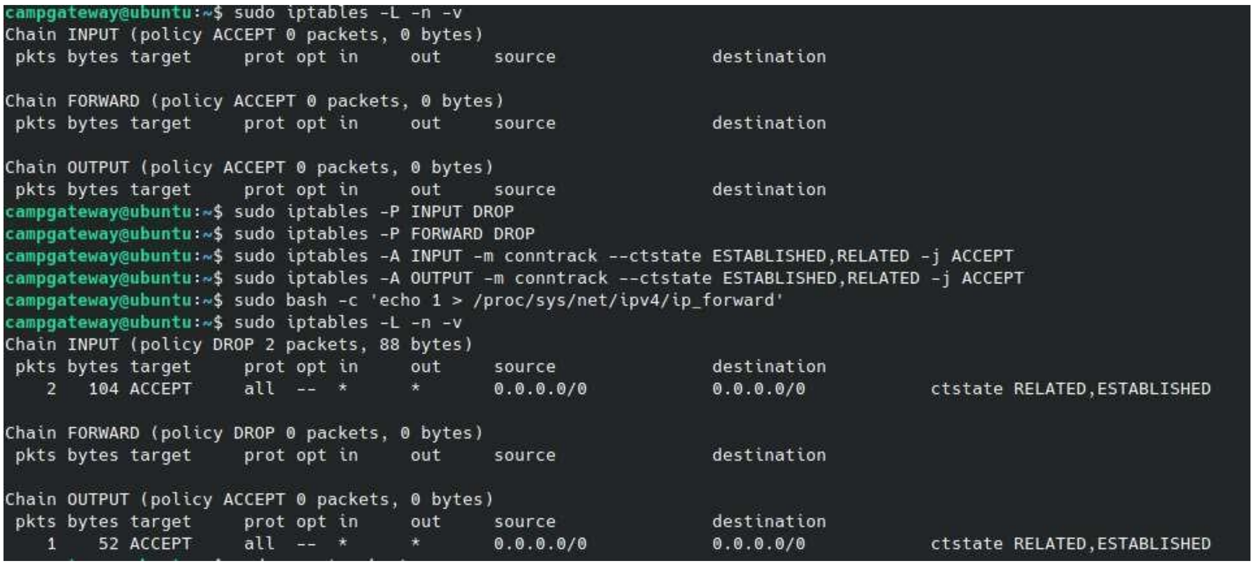


```
camp@ubuntu:~$ sudo /opt/lampp/lampp start
[sudo] password for camp:
Starting XAMPP for Linux 8.0.8-1...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
camp@ubuntu:~$ cd /home/camp/Desktop/SDPcontroller-master/
camp@ubuntu:~/Desktop/SDPcontroller-master$ sudo node ./sdpController.js
SDP Controller running at port 50000
```

Figure 29: Starting the controller

B- Gateway

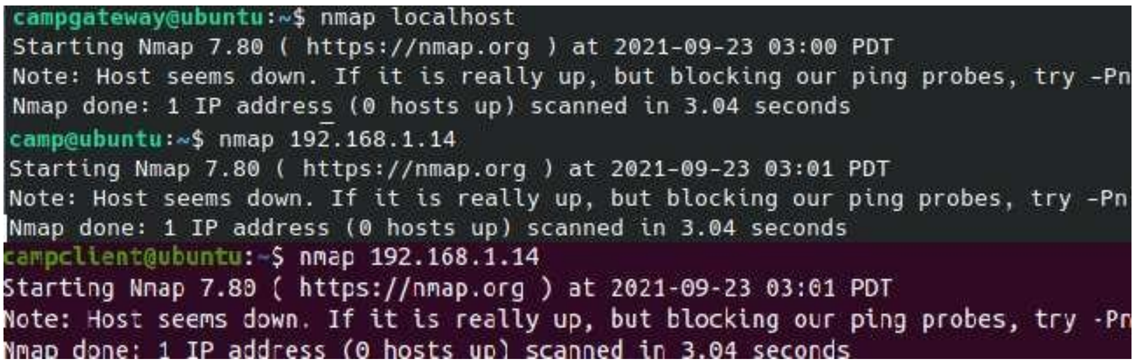
Establishing a drop all policy for input and forward traffic, ensuring that connection tracking is enabled and traffic for established connections is permitted as well as enabling traffic forwarding.



```
campgateway@ubuntu:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
campgateway@ubuntu:~$ sudo iptables -P INPUT DROP
campgateway@ubuntu:~$ sudo iptables -P FORWARD DROP
campgateway@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
campgateway@ubuntu:~$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
campgateway@ubuntu:~$ sudo bash -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
campgateway@ubuntu:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 2 packets, 88 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   104 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    52 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
```

Figure 30: Iptables rules

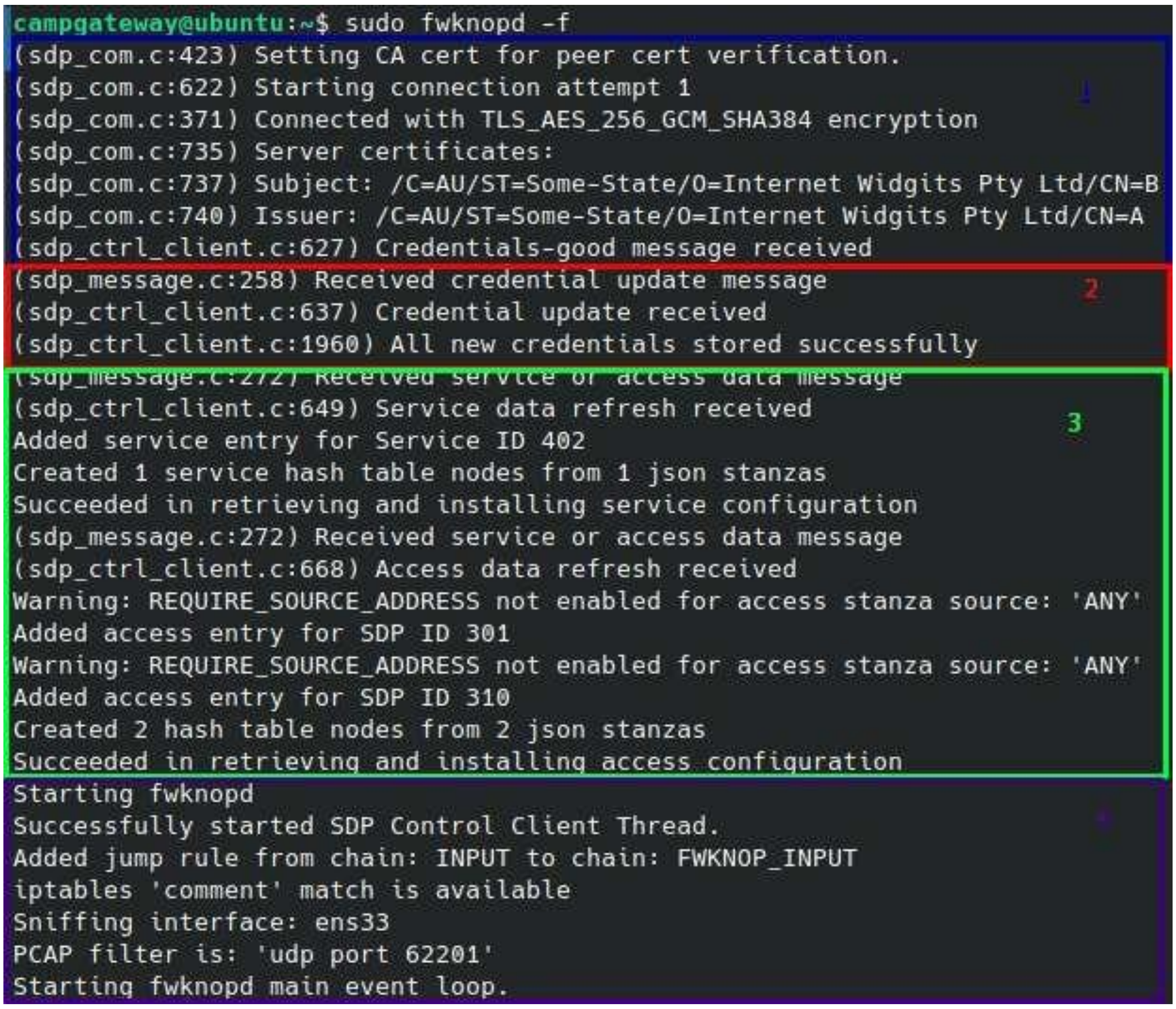
Testing with Nmap to discover hosts and services that are running on the gateway:



```
campgateway@ubuntu:~$ nmap localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-23 03:00 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
camp@ubuntu:~$ nmap 192.168.1.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-23 03:01 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
campclient@ubuntu:~$ nmap 192.168.1.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-23 03:01 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
```

Figure 31: Scan using Nmap

Starting the gateway, if the controller is running and accessible, the gateway should connect and retrieve all necessary data to operate.



```
campgateway@ubuntu:~$ sudo fwknopd -f
(sdp_com.c:423) Setting CA cert for peer cert verification.
(sdp_com.c:622) Starting connection attempt 1
(sdp_com.c:371) Connected with TLS_AES_256_GCM_SHA384 encryption
(sdp_com.c:735) Server certificates:
(sdp_com.c:737) Subject: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=B
(sdp_com.c:740) Issuer: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=A
(sdp_ctrl_client.c:627) Credentials-good message received
(sdp_message.c:258) Received credential update message
(sdp_ctrl_client.c:637) Credential update received
(sdp_ctrl_client.c:1960) All new credentials stored successfully
(sdp_message.c:272) Received service or access data message
(sdp_ctrl_client.c:649) Service data refresh received
Added service entry for Service ID 402
Created 1 service hash table nodes from 1 json stanzas
Succeeded in retrieving and installing service configuration
(sdp_message.c:272) Received service or access data message
(sdp_ctrl_client.c:668) Access data refresh received
Warning: REQUIRE_SOURCE_ADDRESS not enabled for access stanza source: 'ANY'
Added access entry for SDP ID 301
Warning: REQUIRE_SOURCE_ADDRESS not enabled for access stanza source: 'ANY'
Added access entry for SDP ID 310
Created 2 hash table nodes from 2 json stanzas
Succeeded in retrieving and installing access configuration
Starting fwknopd
Successfully started SDP Control Client Thread.
Added jump rule from chain: INPUT to chain: FWKNOP_INPUT
iptables 'comment' match is available
Sniffing interface: ens33
PCAP filter is: 'udp port 62201'
Starting fwknopd main event loop.
```

Figure 32: Starting the gateway

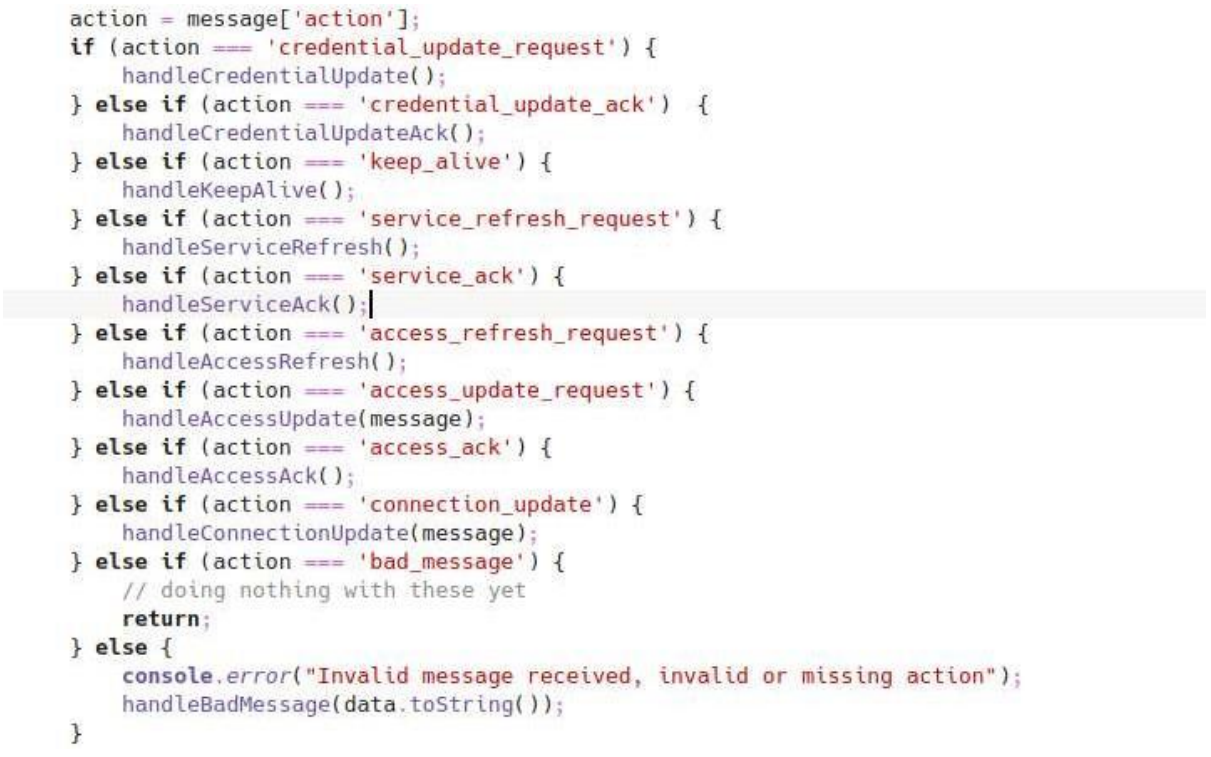
Checking if the gateway spa keys match the existing one the Controller's DB (1).

Updating gateway spa keys both on the DB and on the gateway config file after each successful connection (2).

Retrieving all the services that are concealed under the gateway and the client's information that have access to the specific services (3).

Starting the fwknopd main loop and waiting to add rules to the iptables once a client connection arrives (4).

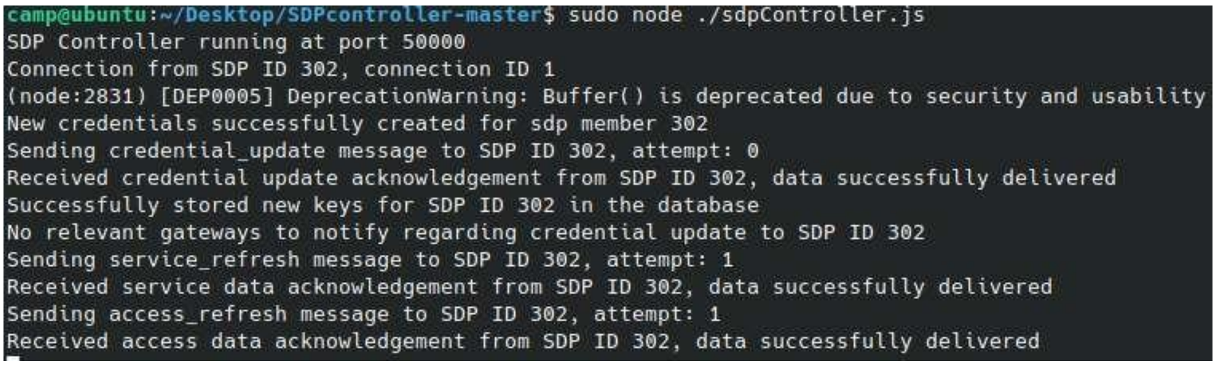
On the other end the controller console reacts in response to the actions sent by the fwknopd server, and reacts accordingly as shown in figure 33.



```
action = message['action'];
if (action === 'credential_update_request') {
    handleCredentialUpdate();
} else if (action === 'credential_update_ack')  {
    handleCredentialUpdateAck();
} else if (action === 'keep_alive') {
    handleKeepAlive();
} else if (action === 'service_refresh_request') {
    handleServiceRefresh();
} else if (action === 'service_ack') {
    handleServiceAck();
} else if (action === 'access_refresh_request') {
    handleAccessRefresh();
} else if (action === 'access_update_request') {
    handleAccessUpdate(message);
} else if (action === 'access_ack') {
    handleAccessAck();
} else if (action === 'connection_update') {
    handleConnectionUpdate(message);
} else if (action === 'bad_message') {
    // doing nothing with these yet
    return;
} else {
    console.error("Invalid message received, invalid or missing action");
    handleBadMessage(data.toString());
}
```

Figure 33: Controller handling Gateway requests

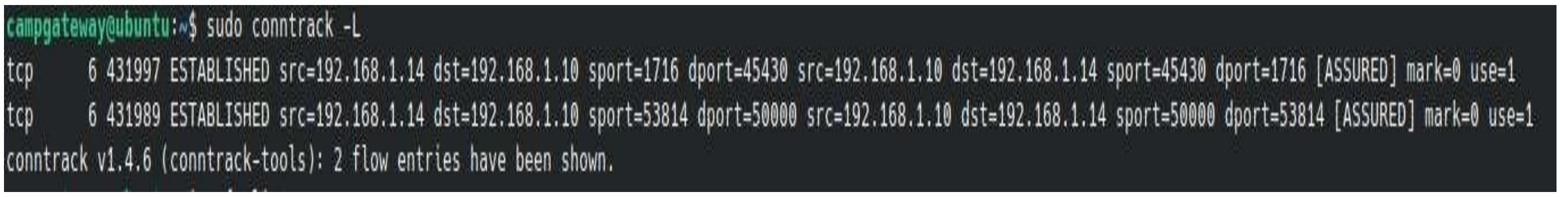
Now that the gateway and controller connection has been established like the figure 34 show:



```
camp@ubuntu:~/Desktop/SDPcontroller-master$ sudo node ./sdpController.js
SDP Controller running at port 50000
Connection from SDP ID 302, connection ID 1
(node:2831) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability
New credentials successfully created for sdp member 302
Sending credential_update message to SDP ID 302, attempt: 0
Received credential update acknowledgement from SDP ID 302, data successfully delivered
Successfully stored new keys for SDP ID 302 in the database
No relevant gateways to notify regarding credential update to SDP ID 302
Sending service_refresh message to SDP ID 302, attempt: 1
Received service data acknowledgement from SDP ID 302, data successfully delivered
Sending access_refresh message to SDP ID 302, attempt: 1
Received access data acknowledgement from SDP ID 302, data successfully delivered
```

Figure 34: Connection between Gateway and Controller

An established connection will be added to the firewall to allow data exchange between both modules.

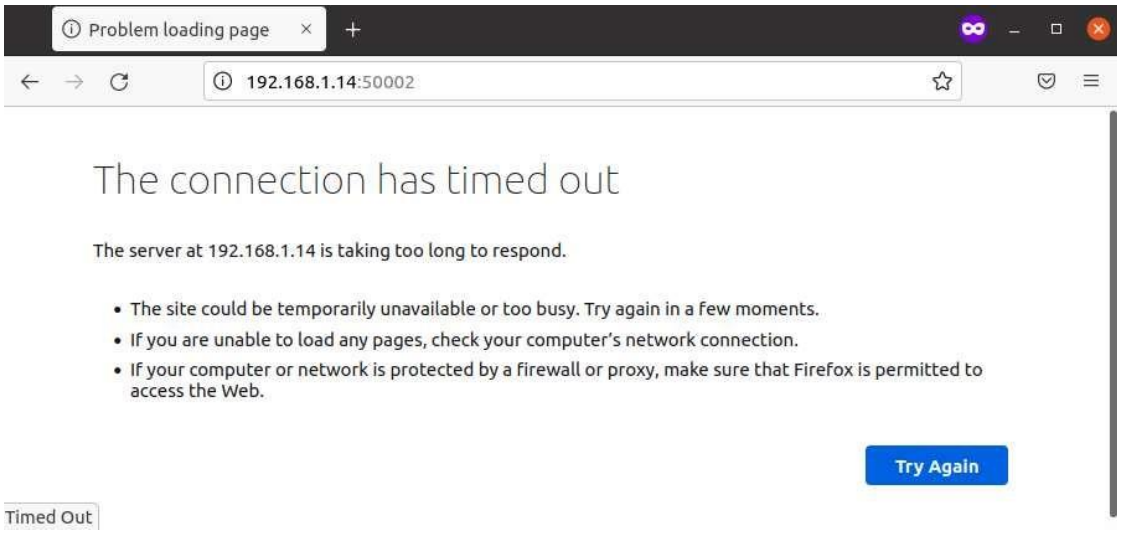


```
campgateway@ubuntu:~$ sudo conntrack -L
tcp      6 431997 ESTABLISHED src=192.168.1.14 dst=192.168.1.10 sport=1716 dport=45430 src=192.168.1.10 dst=192.168.1.14 sport=45430 dport=1716 [ASSURED] mark=0 use=1
tcp      6 431989 ESTABLISHED src=192.168.1.14 dst=192.168.1.10 sport=53814 dport=50000 src=192.168.1.10 dst=192.168.1.14 sport=50000 dport=53814 [ASSURED] mark=0 use=1
conntrack v1.4.6 (conntrack-tools): 2 flow entries have been shown.
```

Figure 35: Established Connections

C- Client

As a client, trying to access the service before authentication is impossible.



Problem loading page

192.168.1.14:50002

The connection has timed out

The server at 192.168.1.14 is taking too long to respond.

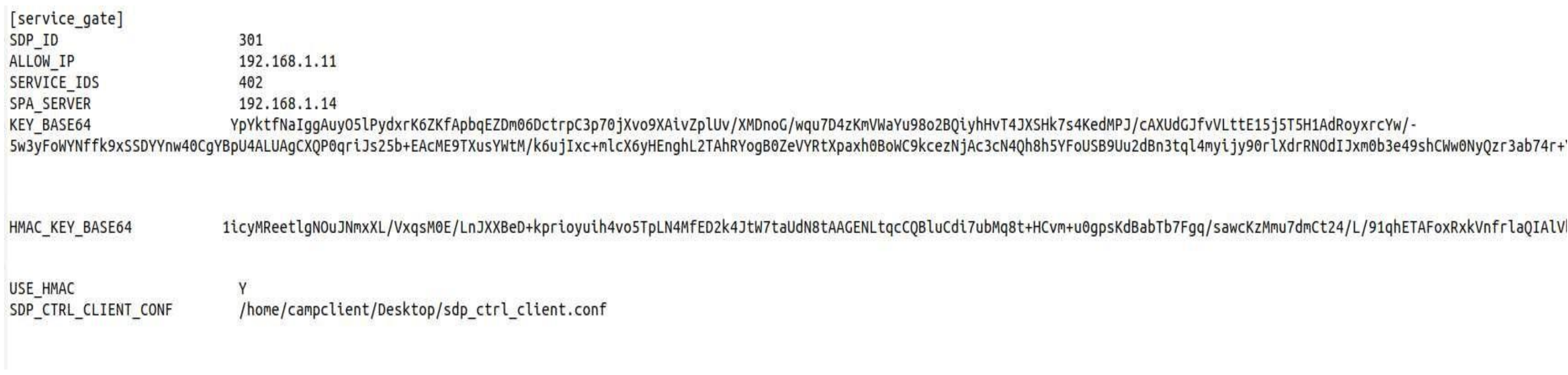
- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again

Timed Out

Figure 36: Unauthorized connection

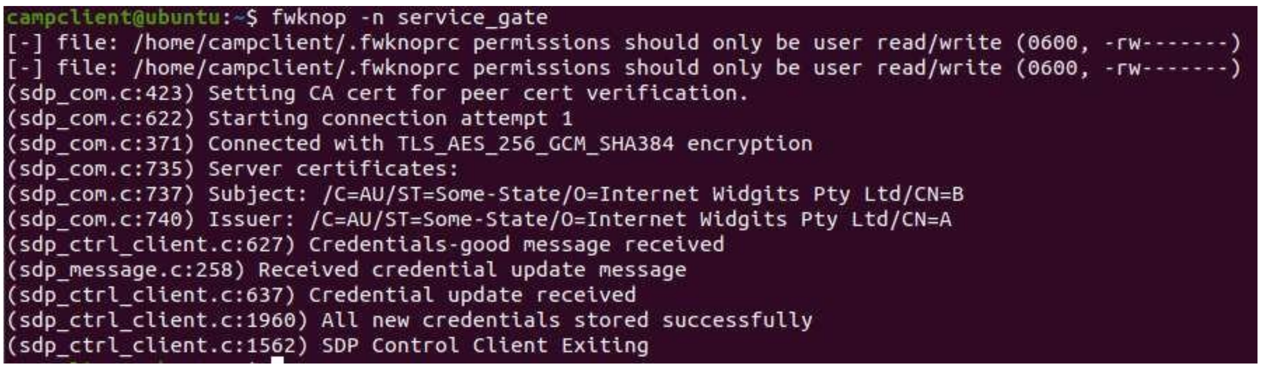
The client executes fwknop module with the following configuration as an input:



```
[service_gate]
SDP_ID                    301
ALLOW_IP                  192.168.1.11
SERVICE_IDS               402
SPA_SERVER                192.168.1.14
KEY_BASE64               YpYktfNaIggAuyO5lPydxrK6ZKfApbqEZDm06DctrpC3p70jXvo9XAivZplUv/XMDnoG/wqu7D4zKmVWaYu98o2BQiyhHvT4JXSHk7s4KedMPJ/cAXUdGJfvVLttE15j5T5H1AdRoyxrcYw/-
5w3yFoWYNffk9xSSDYYnw40CgYBpU4ALUAgCXQP0qriJs25b+EAcME9TXusYWtM/k6ujIxc+mlcX6yHEnghL2TAhRYogB0ZeVYRtXpaxh0BoWC9kcezNjAc3cN4Qh8h5YFoUSB9Uu2dBn3tql4myijy90rlXdrRNOdIJxm0b3e49shCWw0NyQzr3ab74r+


HMAC_KEY_BASE64         1icyMReetlgNOuJNmxXL/VxqsM0E/LnJXXBeD+kprioyuih4vo5TpLN4MfED2k4JtW7taUdN8tAAGENLtqcCQBluCdi7ubMq8t+HCvm+u0gpsKdBabTb7Fgq/sawcKzMmu7dmCt24/L/91qhETAFoxRxkVnfrlaQIAlV


USE_HMAC                  Y
SDP_CTRL_CLIENT_CONF      /home/campclient/Desktop/sdp_ctrl_client.conf
```

Figure 37: Fwknop configuration

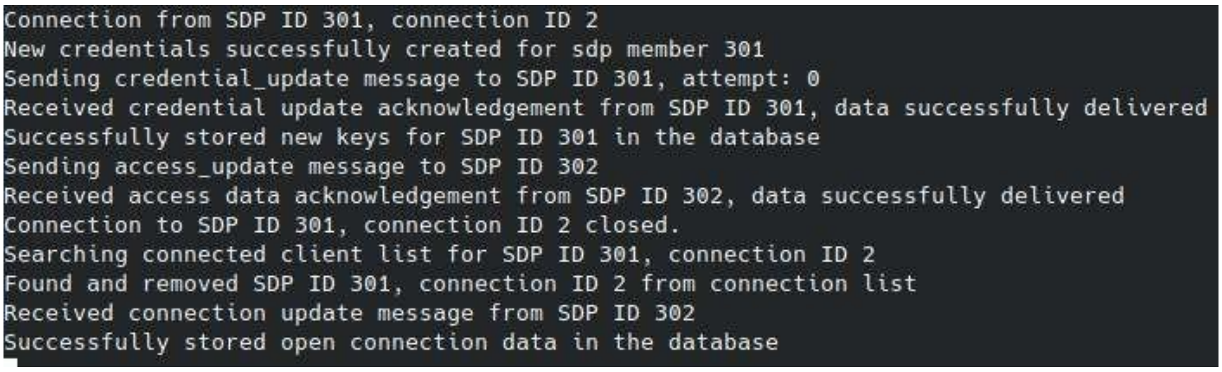
As explained earlier the client uses this stanza to connect to the gateway that is in charge of the requested service, and the output is as follows:



```
campclient@ubuntu:~$ fwknop -n service_gate
[-] file: /home/campclient/.fwknoprc permissions should only be user read/write (0600, -rw-------)
[-] file: /home/campclient/.fwknoprc permissions should only be user read/write (0600, -rw-------)
(sdp_com.c:423) Setting CA cert for peer cert verification.
(sdp_com.c:622) Starting connection attempt 1
(sdp_com.c:371) Connected with TLS_AES_256_GCM_SHA384 encryption
(sdp_com.c:735) Server certificates:
(sdp_com.c:737) Subject: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=B
(sdp_com.c:740) Issuer: /C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=A
(sdp_ctrl_client.c:627) Credentials-good message received
(sdp_message.c:258) Received credential update message
(sdp_ctrl_client.c:637) Credential update received
(sdp_ctrl_client.c:1960) All new credentials stored successfully
(sdp_ctrl_client.c:1562) SDP Control Client Exiting
```

Figure 38: Client connexion

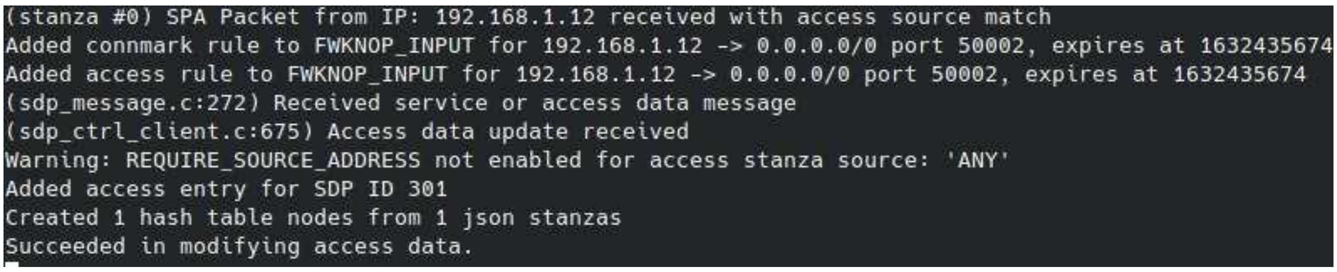
Upon connection the controller renews the client's spa keys and stores them on both sides, and sends an access data acknowledgment to the gateway regarding the client:



```
Connection from SDP ID 301, connection ID 2
New credentials successfully created for sdp member 301
Sending credential_update message to SDP ID 301, attempt: 0
Received credential update acknowledgement from SDP ID 301, data successfully delivered
Successfully stored new keys for SDP ID 301 in the database
Sending access_update message to SDP ID 302
Received access data acknowledgement from SDP ID 302, data successfully delivered
Connection to SDP ID 301, connection ID 2 closed.
Searching connected client list for SDP ID 301, connection ID 2
Found and removed SDP ID 301, connection ID 2 from connection list
Received connection update message from SDP ID 302
Successfully stored open connection data in the database
```

Figure 39: Controller authenticate client

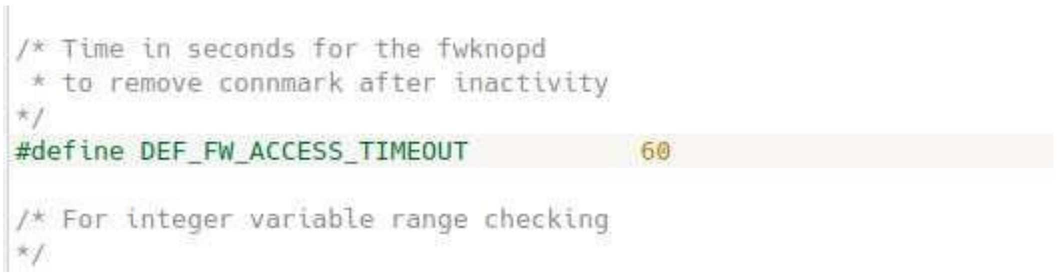
After authentication and spa keys recognition the gateway creates an access rule and establish connection for the client ip address:



```
(stanza #0) SPA Packet from IP: 192.168.1.12 received with access source match
Added connmark rule to FWKNOP_INPUT for 192.168.1.12 -> 0.0.0.0/0 port 50002, expires at 1632435674
Added access rule to FWKNOP_INPUT for 192.168.1.12 -> 0.0.0.0/0 port 50002, expires at 1632435674
(sdp_message.c:272) Received service or access data message
(sdp_ctrl_client.c:675) Access data update received
Warning: REQUIRE_SOURCE_ADDRESS not enabled for access stanza source: 'ANY'
Added access entry for SDP ID 301
Created 1 hash table nodes from 1 json stanzas
Succeeded in modifying access data.
```

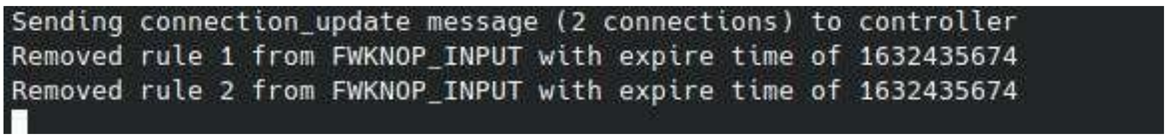
Figure 40: access rule for the client

After 60seconds the gateway removes the access rule



```
/* Time in seconds for the fwknopd
 * to remove connmark after inactivity
*/
#define DEF_FW_ACCESS_TIMEOUT          60

/* For integer variable range checking
*/
```

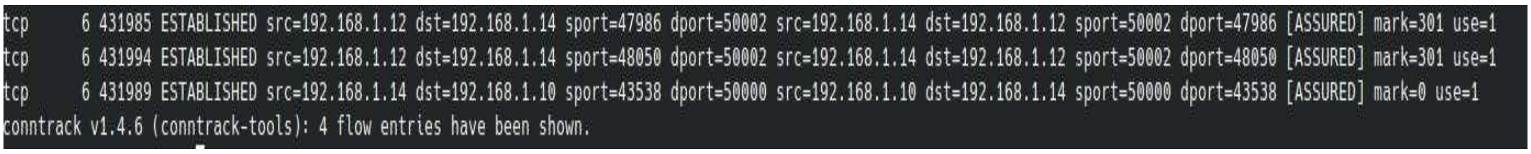
Figure 41: Gateway configuration



```
Sending connection_update message (2 connections) to controller
Removed rule 1 from FWKNOP_INPUT with expire time of 1632435674
Removed rule 2 from FWKNOP_INPUT with expire time of 1632435674
```

Figure 42: Removing access rules from iptables

And the client keeps his access to the service alive because the connections state is established:



```
tcp      6 431985 ESTABLISHED src=192.168.1.12 dst=192.168.1.14 sport=47986 dport=50002 src=192.168.1.14 dst=192.168.1.12 sport=50002 dport=47986 [ASSURED] mark=301 use=1
tcp      6 431994 ESTABLISHED src=192.168.1.12 dst=192.168.1.14 sport=48050 dport=50002 src=192.168.1.14 dst=192.168.1.12 sport=50002 dport=48050 [ASSURED] mark=301 use=1
tcp      6 431989 ESTABLISHED src=192.168.1.14 dst=192.168.1.10 sport=43538 dport=50000 src=192.168.1.10 dst=192.168.1.14 sport=50000 dport=43538 [ASSURED] mark=0 use=1
conntrack v1.4.6 (conntrack-tools): 4 flow entries have been shown.
```

Figure 43: Established connexion list

Client has full access of the service:

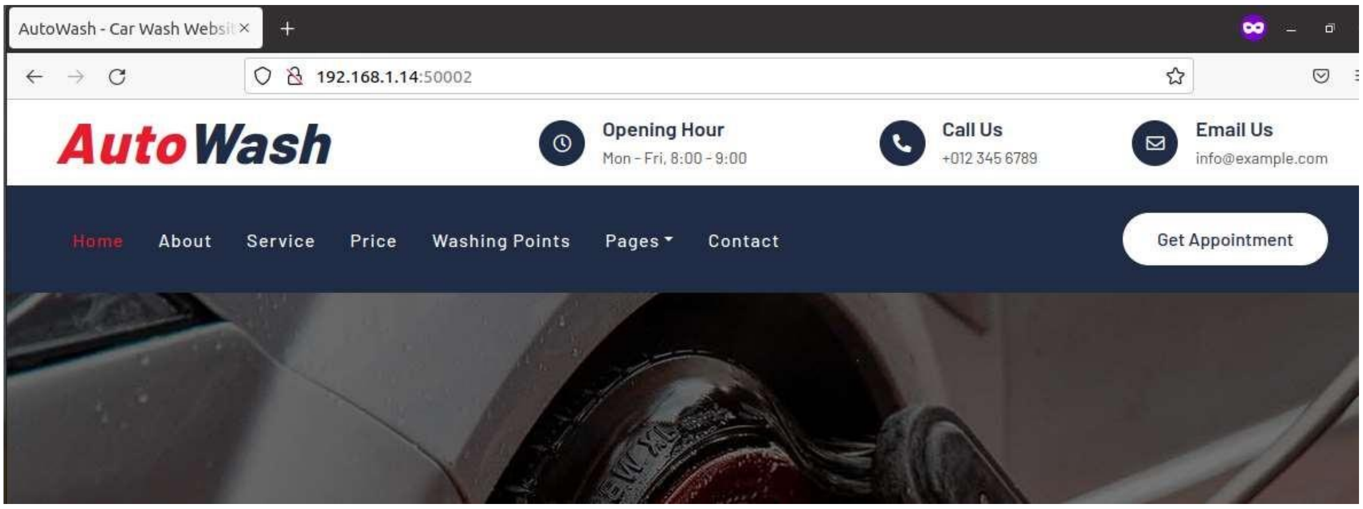
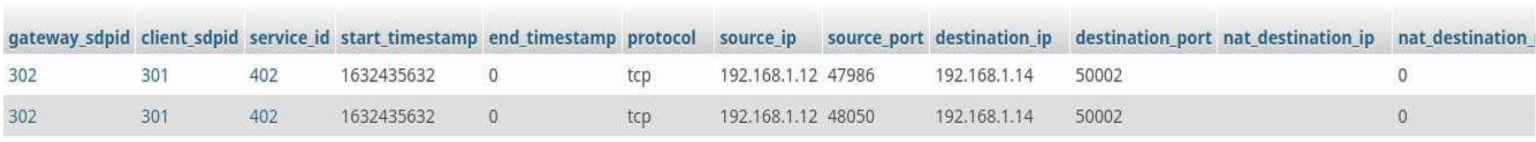


Figure 44: Client connections to the protected service

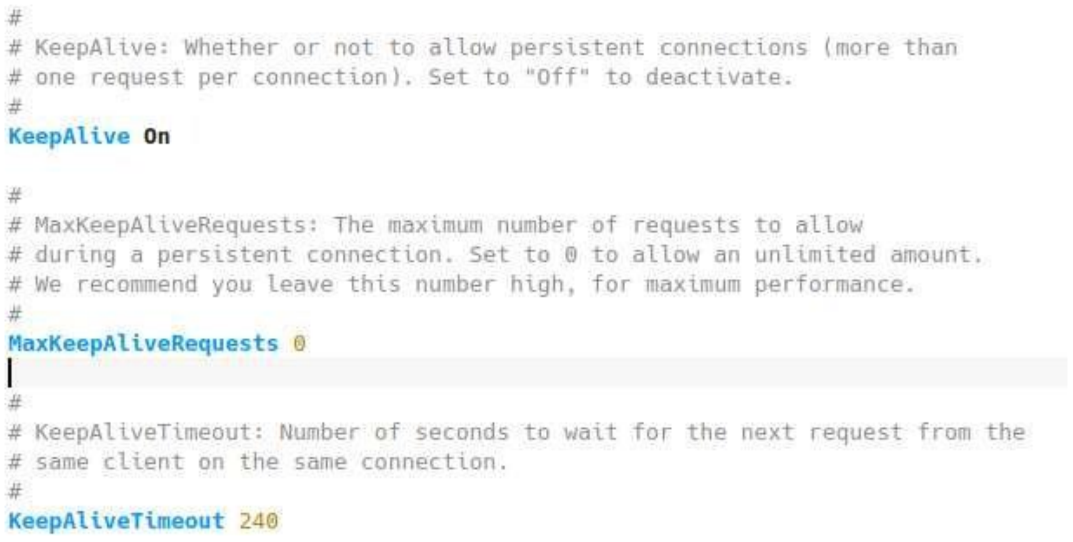
The controller keeps the client's connection open granted he is active on the service Open_Connection table rows:



gateway_sdpid	client_sdpid	service_id	start_timestamp	end_timestamp	protocol	source_ip	source_port	destination_ip	destination_port	nat_destination_ip	nat_destination_
302	301	402	1632435632	0	tcp	192.168.1.12	47986	192.168.1.14	50002		0
302	301	402	1632435632	0	tcp	192.168.1.12	48050	192.168.1.14	50002		0

Figure 45: Open connections table

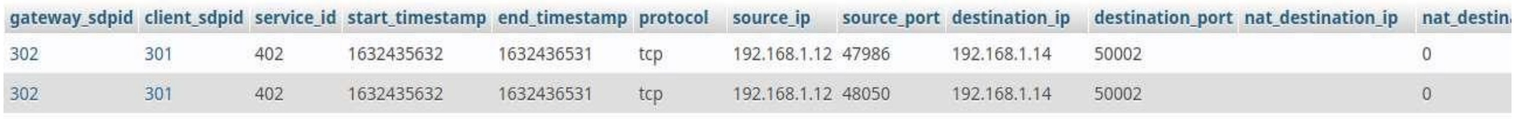
After 240seconds of inactivity from the client on the service the controller closes connection and blocks the client’s ip address until another connection will be established as the configuration shows in figure 46.



```
#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 0

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 240
```

Figure 46: Open connections configuration

Closed_Connection table on the controller



gateway_sdpid	client_sdpid	service_id	start_timestamp	end_timestamp	protocol	source_ip	source_port	destination_ip	destination_port	nat_destination_ip	nat_destin
302	301	402	1632435632	1632436531	tcp	192.168.1.12	47986	192.168.1.14	50002		0
302	301	402	1632435632	1632436531	tcp	192.168.1.12	48050	192.168.1.14	50002		0

Figure 47: Closed Connections Table

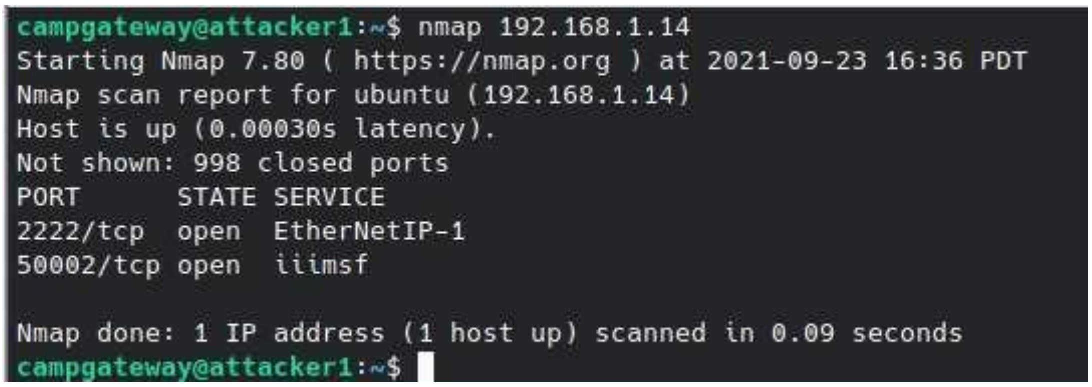
Syn Flood Attack (DOS) using Hping3 and VnStat

- hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols.
  The tool hping3 send manipulated packets, to control the size, quantity and fragmentation of packets in order to overload the target and bypass or attack firewalls. Hping3 can be useful for security or capability testing purposes, using it to test firewalls effectivity and if a server can handle a big number of packets.

- vnStat is a network utility for the Linux operating system. It uses a command line interface. vnStat command is a console-based network traffic monitor. It keeps a log of hourly, daily and monthly network traffic for the selected interface.

By deploying a 4th Virtual Machine acting as an attacker using hping3 tool we were able to test our Black cloud with synflood and monitor the network traffic using vnStat.

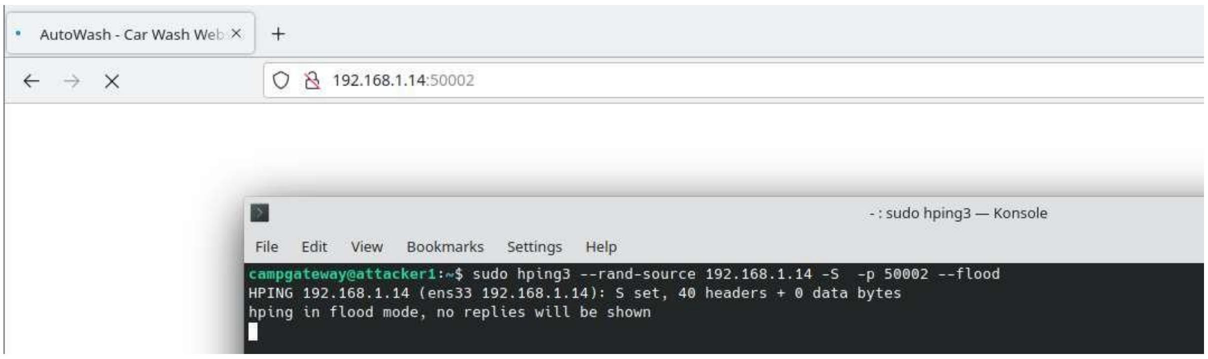
Checking with nmap without running the SDP.



```
campgateway@attacker1:~$ nmap 192.168.1.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-23 16:36 PDT
Nmap scan report for ubuntu (192.168.1.14)
Host is up (0.00030s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
2222/tcp  open  EtherNetIP-1
50002/tcp open  iiimsf

Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
campgateway@attacker1:~$
```

Figure 48: Scanning ports

Initiating synflood stops the services from working and everyone loses access:

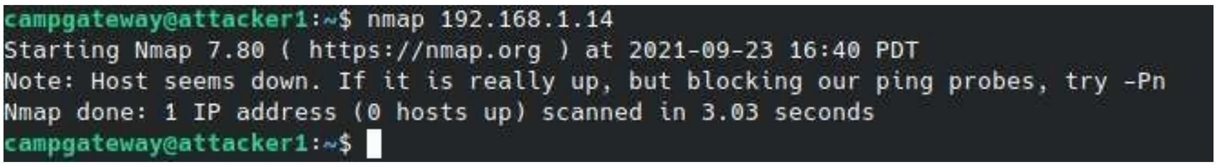


AutoWash - Car Wash Web
192.168.1.14:50002
- : sudo hping3 — Konsole
File Edit View Bookmarks Settings Help

```
campgateway@attacker1:~$ sudo hping3 --rand-source 192.168.1.14 -S  -p 50002 --flood
HPING 192.168.1.14 (ens33 192.168.1.14): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 49: SynFlood Attack

However, this time, the SDP protects the hosts, and does not allow packets from unauthorized users to enter the SDP system. In order to prove its efficiency, vnStat and nmap were used:

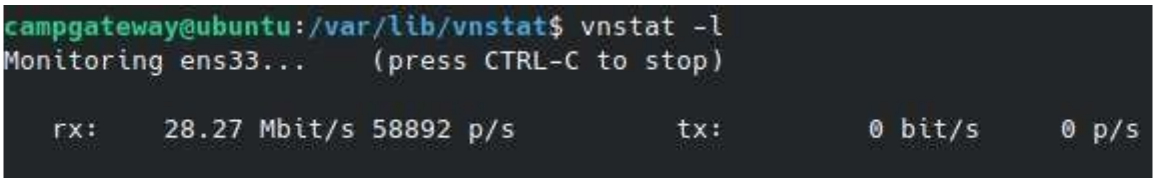


```
campgateway@attacker1:~$ nmap 192.168.1.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-23 16:40 PDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.03 seconds
campgateway@attacker1:~$
```

Figure 50: Nmap scan after initiating sdp

And after attacking with synflood the client keeps connection to the service since iptables drop all policy is blocking everything.

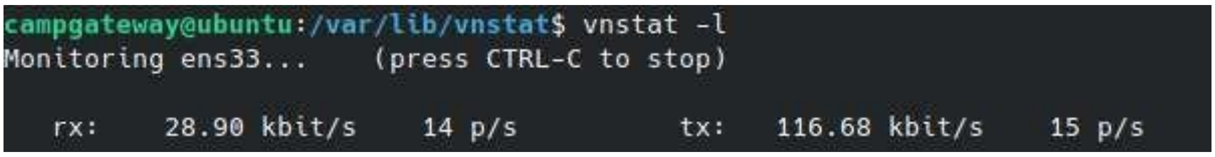
And our network traffic monitor shows heavy traffic on the interface:



```
campgateway@ubuntu:/var/lib/vnstat$ vnstat -l
Monitoring ens33...    (press CTRL-C to stop)

   rx:     28.27 Mbit/s 58892 p/s          tx:         0 bit/s      0 p/s
```

Figure 51: Traffic while the attack

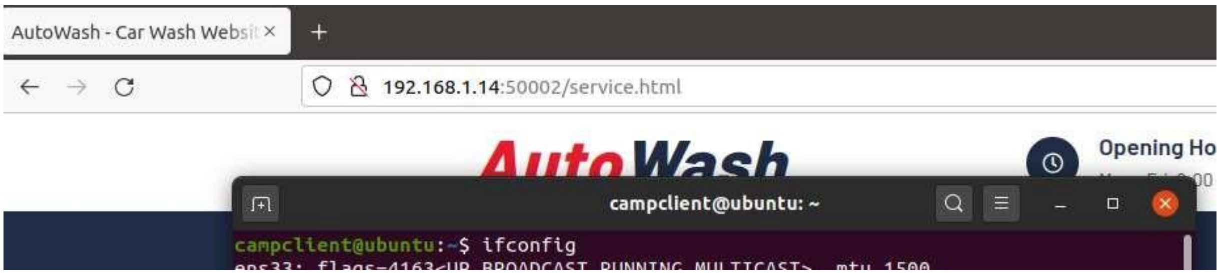
In comparison to without the synflood attack:



```
campgateway@ubuntu:/var/lib/vnstat$ vnstat -l
Monitoring ens33...    (press CTRL-C to stop)

   rx:     28.90 kbit/s     14 p/s          tx:    116.68 kbit/s     15 p/s
```

Figure 52: Traffic in regular conditions

But the client keeps using the service without any difficulties:



```
campclient@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

Figure 53: Service available during the attack

Conclusion

The attack was unsuccessful, since the SDP system blocked it, and the hosts are protected. Most importantly, communication can still take place inside the system, while the clients and the service are protected.

This example further proves the effectiveness of the SDP.

Conclusion and future work

• Conclusion

The primary aim of this project was to implement/discuss the SDP and SPA protocols and mention their benefits. In general, both protocols provide an extra security layer for the services they protect, which means that an attacker would have to penetrate and exploit multiple security mechanisms in order to inflict damage to its target. The SPA protocol provides authentication prior to connection: clients can be authorized to a server without the need to establish a connection with it; therefore, the server can block all its ports via the firewall and allow access to them only to authenticated clients. The SDP on the other hand, uses the SPA to authenticate its clients (IHs) and then provides them with gateways (AHs) to services hidden behind its structure.

SDP provides a simple and user-centric security solution instead of network or data-centric solutions. In an organization, not everyone can see all of the network resources; instead, he/she can only see the resources made available for him/her. Since network resources are not visible to outsiders, these acts as a significant benefit. As SDP adopts a technique to authenticate first and then communicate, users never get the chance to assess properties of security without being authenticated first.

Therefore, attackers have very limited information for network-based attacks to be successful.

• Future work

However, the final implementation has plenty of room for improvements and new ideas. Since the system was built with future implementations and extendibility strongly into consideration, new features are fairly easy to implement and program using the existing mechanisms and structures.

As far as the current implementation is concerned, the client can connect and request access to any service on the SDP system. However, a future step would be to personalize this process, so that every client is linked only with specific services, and has its own profile inside the system, thus building the SDP platform (in the form of an application or a website). Through it, and by using the SDP Catalogue Services, the user would be able to register news services and components to the system, see and communicate with already existing services and customize its profile in a user-friendly manner.

Appendix A
[37]

SDP Packets:

1-Login

The first packet to be exchanged through the LOGIN packet, AH is authorized, and a new session is created by sCTL. E_AID is decrypted using the users' shared secret key and then compared to the AID field in order to authenticate user. The packet is displayed on table 7. Upon registration to the sCTL, the AH also creates its own firewall. Like the sCTL, it drops all incoming connections and packets except those coming from the Controller. Any client that wishes to communicate with the AH must authenticate itself via SPA.

Table 7: Login packet

Field	Value
OP LENGTH DATA	0x01 64 AID/E_AID

2-Login_Resp

The response to LOGIN packet contains the identification number of the created session as well as the maximum time in msec from the last communication between the AH and the sCTL until the connection is closed (keepAlive value). The packet is displayed on table 8.

3-Logout

This message is sent, either by the Controller to the AH or vice versa, in order to indicate the end of the communication. Whoever gets this message first assumes that the connection is over and terminates the session. The packet contains only the LOGOUT packet code: 0x03. The logout process can be seen on figure 54.

Table 8: Login_Resp packet

Field	Value
OP LENGTH DATA	0x02 64 SESSIONID/KEEP_ALIVE

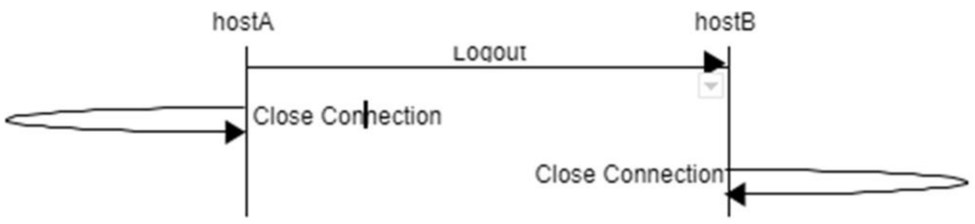


Figure 54: logout procedure

4-Keep-Alive

Indicates that the client is alive. This packet is sent periodically by both members of the connection to keep the session open. In case of delay, the Controller or the AH terminates the connection (as seen on figure 55). The packet contains only the KEEP_ALIVE packet code which is 0x04

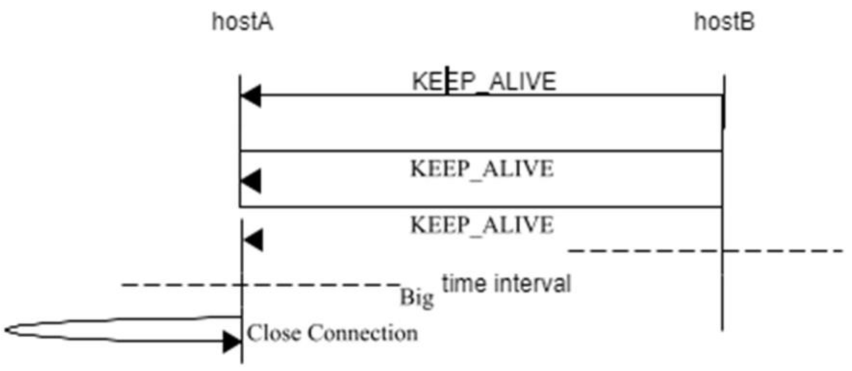


Figure 55: Keep-Alive periodical send

5-AH-Service

This packet is sent by sCTL to AH to inform it about the list of services it is going to provide. The list is formatted as a json array which contains the following fields:

- id: the id of the service.
- name: the name of the service.
- ip: the address of the service.
- port: the port of the service.
- type: the type of service (http, tcp, etc.). This instructs the AH how to connectwith the service.

When the list is received, the AH adds rules to the firewall allowing communication with these services. The packet structure is displayed on table 9.

Table 9: AH-Service packet

Field	Value
OP LENGTH DATA	0x06 LEN SERVICES_JSON

6-IH-AUTH

Sent by the sCTL to the AH when an IH user has been successfully authenticated and wants to use a service that this AH provides. The IH info is stored in a json object with the following fields:

- aid: the aid of the IH
- seed: the seed to use in the connection with the IH
- service_id: the id of the service requested

As soon as this message is received, the AH is ready to accept SPA packets from the IH. The packet is depicted on table 10.

Table 10: IH-AUTH Packet

Field	Value
OP LENGTH DATA	0x07 LEN IH_INFO_JSON

7-Login

The first packet to be exchanged. Using a LOGIN packet, the IH is authorized and anew session is created by the Controller. The LOGIN packet has already been displayed on table 7.

8-Login-Resp

The response to LOGIN packet contains the identification number of the new session. It is shown on table 11.

Table 11: Login-Resp Packet

Field	Value
OP	0x02
LENGTH	64
DATA	SESSION_ID

In contrast with the AH-Controller LOGIN_RESP, the KEEP_ALIVE field is not used here. sCTL terminates connection after a predefined period if the IH is inactive or when the IH has found an available AH to connect with.

9-Logout

This message is sent by the Controller to the IH or vice versa in order to indicate the end of communication. Whoever gets this message assumes that the connection is over and terminates the session. This packet is exactly the same with the LOGOUT packet in AH-sCTLcommunication.

10-IH-Query

When an IH has finally logged in, it can ask the controller to provide it with either a full set of services that it can use, or a subset thereof, by submitting a specific query.

The query is a json object and its fields can be seen below:

- name: the name of the service.
- type: the type of the service.
- service_id: the ID of the service.

When the IH is sure about the service it intends to use, it sends an IH_QUERY to the sCTL setting the service_id field with only the ID of the corresponding service. The sCTL then starts the procedure of connecting the IH with the designated AH. The packet is depicted on table 12.

11-IH-Service

After an IH_QUERY packet has been received, the IH_SERVICES packet is sent by the sCTL to the IH in order to inform the latter about the list of services matching the IH's query. The services list is a json object whose fields are described bellow.

Table 12: IH-Query packet

Field	Value
OP LENGTH DATA	0x08 LEN QUERY_JSON

- name: the name of the service
- type: the type of the service
- service_id: the ID of the service

The packet structure is shown on table 13.

Table 13: Ih-Service Packet

Field	Value
OP LENGTH DATA	0x06 LEN SERVICES_JSON

12-AH-Ready

After receiving the IH_SERVICES packet with the service_id field set, the sCTL selects an appropriate AH to handle the request and informs it about the new IH through an IH_AUTH packet. When AH has acknowledged the AH_AUTH, the sCTL sends an AH_READY packet to the IH to inform it about its new gateway to the service. The packet includes information about the AH in json format, which is described below:

- ip: the IP address of the AH.

- seed: the seed to be used during the communication.
- service_id: the ID of the service that it is going to provide the AH_READY packetformat is depicted on table 14.

13-LOG-ACK

A client sends this to the server to acknowledge the receipt of the LOGIN_RESP packet. The packet is sent once the client has initialized a session. Once the server receives this message, it sets the clients session to active. The packet's code is 0x10.

Table 14: AH-Ready packet

Field	Value
OP LENGTH DATA	0x09 LEN AH_INFO_JSON

14-AHS-ACK

The AH sends this ACK to the sCTL in order to acknowledge the receipt of the AH_SERVICES packet. It is sent once the AH allows the new services to connect with it via its firewall. After receiving this packet, the sCTL can assign IHs to this AH for the services it provides. The packet's code is 0x11.

15-IHA-ACK

The AH sends this ACK to Controller in order to acknowledge the receipt of the IH_AUTH packet. It is only sent once the AH has updated its configuration files with client-related data. When this packet is received, the sCTL sends the AH_READY packet to the IH . The packet's code is 0x12.

16- OPEN-CONN-REQ

Much like the LOGIN packet, the OPEN_CONN_REQ packet is used by the IH in order to ask AH to connect with it (using the shared seed, exchanged on the service query step, in order to validate). The packet is displayed on table 15. The AH opens a connection with the requested service during the communication. In case the communication fails, it sends an error message to the IH and terminates their connection.

Table 15: OPEN-CONN-REQ Packet

Field	Value
OP LENGTH DATA	0x07 64 AID/E_AID

17- OPEN_CONN_RESP

The OPEN_CONN_RESP packet is used as a response to the OPEN_CONN_REQ packet. It contains the MUX for this session. A MUX is a 32-byte string containing the service_id (First 16 bytes) and the session_id (last 16 bytes). In order to refer to this service, the IH and AH have to use this MUX. The packet structure is displayed on table 16.

Table 16: Open-Conn-Resp Packet

Field	Value
OP LENGTH DATA	0x08 32 MUX

18- Conn-close

The CONN_CLOSE packet ends the connection between the IH and AH. The code of this packet is 0x0A.

19- Data

The DATA packet is service data encapsulated inside an SDP packet, which is either forwarded, via the AH, to either the service or the IH. The packet is displayed on table 17.

Table 17: Data Packet

Field	Value
OP LENGTH DATA	0x09 LEN SERVICE_PAYLOAD

Bibliography

[1] B. Vigliarolo, “Who has banned Zoom? Google, NASA, and more,” 2020. [Online]. Available: https://www.techrepublic.com/article/who-has-banned-zoom-google-nasa-and-more
[2] Information Security (2001) Wikipedia, the Free Encyclopaedia. [Online]. http://en.wikipedia.org/wiki/Information_security
[3] Software Defined Perimeter Working Group-Cloud Security Alliance (CSA), “Software Defined Perimeter,” Dec. 2013. [Online]. http://downloads.cloudsecurityalliance.org/initiatives/sdp/Software Defined Perimeter.pdf
[4] P. Kumar et al., “Performance Analysis of SDP for Secure Internal Enterprises,” Proc. IEEE Wireless Commun. Networking Conf. (WCNC’19), Apr. 2019.
[5] Ahmed, Sheeraz & Khan, Hamayun & Saeed, Khalid. Penetration Testing Active Reconnaissance Phase -Optimized Port Scanning With Nmap Tool. 10.1109/ICOMET.2019.8673520, 2019.
[6] Krzywinski, R. “Port Knocking: Network Authentication Across Closed Ports,” SysAdmin Magazine, vol. 12, June 2003, pp. 12-17.
[7] http://www.portknocking.org/
[8] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. The effect of internet security breach announcements on market value: Capital market reactions for breache firms and internet security developers. International Journal of Electronic Commerce, 9(1):70–104, 2004.
[9] Patel, Nimisha & Patel, Rajan & Patel, Dhiren. (2009). Packet Sniffing: Network Wiretapping. IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6–7 March 2009. 2691-2696.
[10] Vivo, Marco & Ke, Le & Isern, Germinal & Vivo, Gabriela. (1999). A review of port scanning techniques. Computer Communication Review. 29. 41-48. 10.1145/505733.505737.
[11] Toby Ehrenkranz and Jun Li. On the state of ip spoofing defense. ACM Transactions on Internet Technology (TOIT), 9(2):6, 2009.
[12] Yvo Desmedt. Man-in-the-middle attack. In Encyclopedia of cryptography and security pages 759–759. Springer, 2011.
[13] Paul Syverson. A taxonomy of replay attacks [cryptographic protocols]. In Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings, pages 187–191. IEEE, 1994.
[14] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. Internet denial of service:Attack and defense mechanisms (radia perlman computer networking and security). 2004.
[15] Gustavus J Simmons. Symmetric and asymmetric encryption. ACM Computing Surveys (CSUR), 11(4):305–330, 1979.
[16] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. Evaluating the performance of symmetric encryption algorithms. IJ Network Security, 10(3):216–222, 2010.
[17] Shahzadi Farah, Younas Javed, Azra Shamim, and Tabassam Nawaz. An experimental study on performance evaluation of asymmetric encryption algorithms. In Recent Advances in Information Science, Proceeding of the 3rd European Conf. of Computer Science,(EECS-12), pages 121–124, 2012.

[18] Bart Preneel. Cryptographic hash functions. Transactions on Emerging Telecommunications Technologies, 5(4):431–448, 1994.
[19] Russell Housley, Warwick Ford, William Polk, and David Solo. Internet x. 509 public key infrastructure certificate and crl profile. Technical report, 1998.
[20] Eric Rescorla. SSL and TLS: designing and building secure systems, volume 1. Addison Wesley Reading, 2001.
[21] Tatu Ylonen and Chris Lonvick. The secure shell (ssh) connection protocol. 2006.
[22] Loki Radoslav. Ssh file transfer protocol. 2012.
[23] Howard A Seid and ALbert Lespagnol. Virtual private network, June 16 1998. US Patent 5,768,271.
[24] Gordon Fyodor Lyon. Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure, 2009.
[25] James E Smith and Ravi Nair. The architecture of virtual machines. Computer, 38(5):32– 38, 2005.
[26] J. Garbis, P. Thapliyal, B. Flores, and J. Islam, "Software defined perimeter for infrastructure as a service," Cloud Secur. Alliance, Seattle,WA, USA, Tech. Rep., 2016.
[27] M. J. Kaur and P. Maheshwari, "Building smart cities applications using IoT and cloud- based architectures," in Proc. Int. Conf. Ind. Informat. Comput. Syst. (CIICS), Mar. 2016, pp. 1–5.
[28] B. Bilger, "Software defined perimeter working group SDP specification 1.0," Cloud Secur. Alliance, Seattle, WA, USA, Tech. Rep., Apr. 2014.
[29] S. Rose, O. Borchet, S. Mitchel, and S. Connelly, "Zero trust architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 09/23/19: SP 800-207 (1st Draft), 2019.
[30] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in Proc. 16th ACM Conf. Comput. Commun. Secur., 2009,pp. 199–212.
[31] J. Idziorek and M. Tannian, "Exploiting cloud utility models for profit and ruin," in Proc. IEEE 4th Int. Conf. Cloud Comput., Jul. 2011, pp. 33–40.
[32] S. Meena, E. Daniel, and N. A. Vasanthi, "Survey on various data integrity attacks in cloud environment and the solutions," in Proc. Int. Conf. Circuits, Power Comput. Technol. (ICCPCT), Mar. 2013,pp. 1076–1081.
[33] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (SDP): State of the art secure solution for modern networks," IEEE Netw., vol. 33, no. 5, pp. 226–233, Sep. 2019.

Figures

[34] Man in the middle (MITM) attack https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/
[35] Cryptography by Kathleen Richards https://searchsecurity.techtarget.com/definition/cryptography
[36] Cryptography types https://www.encryptionconsulting.com/education-center/what-is-cryptography/
[37] Development of Software Defined Securty Perimeter 2018 Fotios-Dimitrios Tsokos. Supervised by: Lalis Spyros. Antonopoulos Christos. February 2018. Institutional Repository - Library & Information
[38] Software-Defined Perimeter (SDP) State of the Art Secure Solution For Modern Networks by Abdallah Moubayed, Ahmed Refaey and Abdallah Shami