MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

UTILIZACIÓN DE ENCLAVES SEGUROS PARA LA SECURIZACIÓN DEL ACCESO AL DNS

Estudiante: Angulo Manzanas, Asier

Director/Directora: Jacob Taquet, Eduardo

Curso: 2022-2023

Fecha: Bilbao, 18, 05, 2023

PÁGINA EN BLACNO

RESUMEN

Hoy en día, el avance tecnológico constante, la digitalización y el uso de Internet ha generado que toda la información confidencial de los usuarios circule a través de la red.

A consecuencia de este aumento se están produciendo, en un gran porcentaje, ciberataques que sufren las comunicaciones por las que circula toda esta información confidencial, haciendo necesario el desarrollo de mecanismos de seguridad que aumenten su confidencialidad e integridad evitando así ataques como el "Man in the Middle". Existen protocolos de comunicación transparentes al usuario, como DNS, que, actualmente, carecen de estos mecanismos y siguen presentado vulnerabilidades como el envío de la información en texto plano o la falta de autenticación del servidor.

Debido a esta carencia, en este documento se presenta una solución basada en el concepto de los enclaves para dotar de mayores medidas de seguridad al acceso al DNS. Los enclaves consisten en la configuración de Entornos de Ejecución Fiables en donde ejecutar aplicaciones consideradas críticas y guardar la información que manejan dichas aplicaciones.

Palabras clave: Enclaves seguros, Entornos de Ejecución Fiables, DNS, vulnerabilidades, ciberataques, mecanismos de seguridad, programa.

LABURPENA

Gaur egun etengabeko aurrerapen teknologikoak, digitalizazioa eta Interneten erabilera erabiltzaileen informazio konfidentzial guztia saretik hedatua egotea eragin du.

Igoera horren ondorioz, informazio konfidentzial hori mugitzen den komunikazioetan gero eta zibereraso gehiago gertatzen ari dira, konfidentzialtasuna eta osotasuna bermatzen duten segurtasun mekanismoen garapena beharrezkoa eginez, "Man in the Middle" bezalako erasoak saihesteko. Badira erabiltzailearentzako gardenak diren komunikazio-protokoloak, DNS adibidez, gaur egun segurtasun mekanismo horiek ez dituztenak eta ahultasunak aurkezten jarraitzen dituztenak, esate baterako informazioa testu arruntean bidaltzea edo zerbitzariaren egiaztapen eza.

Gabezia hori dela eta, dokumentu honetan DNS-rako sarbideari segurtasun neurri handiagoak emateko enklabeen kontzeptuan oinarritutako irtenbide bat aurkezten da. Enklabeak Exekuzio Ingurune Fidagarriak konfiguratzean datza, kritikotzat kontsideratzen diren aplikazioak exekutatzeko eta aplikazio horiek erabiltzen duten informazioa gordetzeko.

Gako-hitzak: Enclave seguruak, Exekuzio Ingurune Fidagarriak, DNS, ahultasunak, zibereasoak, segurtasun mekanismoak, programa.

ABSTRACT

Nowadays, constant technological progress, digitalisation and the use of Internet have meant that all users' confidential information is transmitted through the network.

As a result of this increase, a large percentage of cyber-attacks are produced by the communications through which all this confidential information circulates, making it necessary to develop security mechanisms that increase their confidentiality and integrity, thus avoiding attacks such as "Man in the Middle". There are communication protocols that are transparent to the user, such as DNS, which currently lack these mechanisms and continue to present vulnerabilities such as sending information in plain text or the lack of server authentication.

Due to this lack, this document presents a solution based on the concept of secure enclaves to provide greater security measures for DNS access systems. Enclaves consist in the configuration of Trusted Execution Environments where lauch applications considered critical and store the information handled by these applications.

Key words: Secure enclave, Trusted Execution Enviroment, DNS, vulnerabilities, cyberattacks, security mechanisms, programme.

Universidad del País Vasco Euskal Herriko Unibertsitatea

TABLA DE CONTENIDO

Universidad del País Vasco Euskal Herriko Unibertsitatea

ÍNDICE DE FIGURAS

Universidad del País Vasco Euskal Herriko Unibertsitatea

ÍNDICE DE TABLAS

ACRÓNIMOS

AC: Autoridad de Certificación

AMD: Memory Encryption Technology

DNS: Domain Name System

DNSSEC: Domain Name System Security Extension

MAC: Message Authentication Code

MET: Memory Encryption Technology

RDP: Remote Desktop Protocol

SDK: software development kit

SEV: Secure Encrypted Virtualization

SGX: Secure Guard Extension

SME: Secure Memory Encryption

TTE: Trusted Execution Enviroment TEE

TLS: Ttansport Layer Security

UML: Unified Modelling Languaje

1 INTRODUCCIÓN

La incorporación de Internet a la vida cotidiana, con un uso intensivo de servicios multimedia y multitud de tecnologías móviles, ha traído consigo un incremento bastante notable del número de usuarios conectados a la red. Sin embargo, también han aumentado de manera notoria, el número de ciberataques que sufren las personas o las empresas en su día a día.

En los últimos años, los niveles de cibercriminalidad se han disparado hasta alcanzar niveles preocupantes. En la siguiente tabla se pueden observar los datos de España recogidos en el Balance de Criminalidad del Cuarto Trimestre de 2022 [1] de los últimos años:

	Año 2019	Año 2021	Año 2022
Criminalidad TOTAL	1.981.173	1.652.242	1.949.852
Cibercriminalidad	218.302	305.477	375.506
Porcentaje	**11%**	**18%**	**19%**

Tabla 1: Infracciones penales registradas en los últimos 4 años [1]

Esta tabla muestra los valores de criminalidad totales en España durante los últimos 4 años, y cuáles de estos están relacionados con la cibercriminalidad. Como se puede observar, estos datos abalan lo comentado respecto a los ciberataques registrados en el código penal, siendo en el año anterior casi el 20% del total de delitos cometidos. Es más, según los datos facilitados por el Balance de criminalidad del primer trimestre de 2023 [2], estos valores han aumentado llegando a ser 107.879 los delitos registrados relacionados con la ciberseguridad, lo que hace que se incremente a un 22% de los delitos totales registrados en el periodo de referencia.

Debido a estos datos tan preocupantes, es necesario aplicar más mecanismos de seguridad, tanto a nivel de empresa como a nivel personal, especialmente en aquellos protocolos que no son transparentes a los usuarios. Uno de estos mecanismos consiste en el cifrado de la información extremo a extremo a través de criptografía simétrica o asimétrica para mejorar la confidencialidad e integridad y así proteger las comunicaciones al exterior de ataques como el de “Man in the Middle” u Hombre en medio.

Sin embargo, a los protocolos o comunicaciones transparentes al usuario, como el protocolo DNS, no se les dota con el mismo grado de seguridad. A pesar de estar implementando técnicas de seguridad en DNS como DNSSEC o mediante la realización de DoT (DNS over TLS) DNS sobre TLS, la mayoría de las consultas DNS se realizan a través del protocolo de transporte UDP, que es vulnerable a ataque de Man in the Middle. Un ataque de este estilo puede manipular el mensaje de respuesta modificando la dirección IP para redirigir al usuario a una página web maliciosa y aprovechar para obtener información sensible del usuario.

Hoy en día, se está instaurando como tendencia la realización de cifrado a nivel de hardware. Estos mecanismos crean Entornos de Ejecución Fiables (TEE Trust Execution Enviroment), que son entornos aislados, llamados enclaves, donde se ejecutan programas enteros o partes de un programa para dotarlo de mecanismo de autenticación de usuario, y garantizar así la confidencialidad y la integridad. La **finalidad principal del presente proyecto ha sido el hacer uso de los enclaves para dotar de mecanismos de seguridad al protocolo DNS** mejorando así la confidencialidad e integridad del protocolo. Además, hemos creado un método en el que un

usuario pueda conectarse al mismo servidor de forma continua para evitar ataques de usurpación de identidad.

Para cumplir con el objetivo, en primer lugar, hemos realizado un Estado del Arte para analizar, exhaustivamente, los diferentes tipos de mecanismos existentes para la creación de Entornos de Ejecución Fiables y realizar una comparación para ver que mecanismo genera un TEE con mayores prestaciones de seguridad.

En segundo lugar, haciendo uso del TEE más completo, se ha realizado el despliegue de un programa para crear un enclave que nos permita securizar el protocolo DNS. Por último, hemos realizado unas pruebas de validación para comprobar el grado de seguridad que ofrecen el elegido/creado.

2 CONTEXTO

Hoy en día, gracias a la digitalización, la tecnología se ha convertido en uno de los cimientos centrales de sociedad y en el día a día de cada persona. Esto ha traído un sinfín de ventajas y comodidades a la vida cotidiana, como por ejemplo, la posibilidad de disponer de una cantidad inmensa de información sobre cualquier tema en cuestión en segundos.

Es tal el impacto generado por la digitalización que podemos decir que, actualmente, toda la información personal y confidencial de una persona está almacenada en formato digital. Información como el DNI de una persona, su número de cuenta bancaria y otros datos sensible se encuentra ya digitalizada. La existencia de sistemas que almacenan y gestionan este tipo de información altamente confidencial, ha generado una necesidad de disponer de mecanismos de seguridad altamente sofisticados, ya que estos datos son el objetivo principal en muchos de los ciberataques.

Existen diferentes mecanismos de seguridad utilizados para la protección de información y la realización del intercambio de la misma de forma segura. Uno de los más usados para la transmisión segura de información segura es el protocolo TLS (Ttansport Layer Security). TLS es un protocolo de la capa de Transporte que proporciona una comunicación, extremo a extremo, cifrada.

Sin embargo, este tipo de comunicaciones no se suelen utilizar en protocolos secundarios como el protocolo de la capa de aplicación DNS, dejándolo vulnerable a ataques del tipo “Man in the Middle” o de Usurpación de Identidad. Para ofrecer seguridad a este protocolo, se ha diseñado el protocolo DNSSEC (Domain Name System Security Extension) el cual tiene como finalidad la inclusión de mecanismos de seguridad a las consultas DNS mediante el cifrado de la información que va transmitiendo de servidor en servidor [35]. Sin embargo, la implementación de este protocolo es bastante compleja y, debido a los mensajes de control que se intercambian en el proceso, los tiempos de procesamiento y transmisión de una petición y respuesta DNS aumentan considerablemente, al igual que el uso de CPU del equipo que participa en las comunicaciones. Esto genera la necesidad de desarrollar un sistema capaz de ofrecer seguridad al protocolo haciendo uso de una menor utilización de recursos de los equipos.

Hoy en día, se han diseñado nuevas técnicas de ciberseguridad basadas en la creación de Entornos Seguros de Ejecución, llamados **enclaves**, que ofrecen cifrado a nivel de hardware. Estos mecanismos proporcionan un cifrado de las páginas de memoria en la que se almacena la información y el programa que se está ejecutando, aislándolo del resto de agentes externos. Solo programas que pasan un riguroso control de acceso son capaces de acceder a la información de los mismos. Debido a esto, los enclaves ofrecen mecanismos de autenticación y autorización, así como confidencialidad e integridad del mensaje, por lo que se vuelve una opción viable a la hora de dotar agentes externos de mecanismos de seguridad.

Es, en este contexto, en el que se enmarca el presente trabajo cuyo objetivo es desarrollar un software que permita dotar de mecanismos de seguridad al protocolo DNS haciendo uso de enclaves seguros.

3 OBJETIVOS Y ALCANCE

3.1 Objetivos

El objetivo principal del presente trabajo de Fin de Máster es la securización del protocolo DNS para asegurar la integridad y confidencialidad del mensaje mediante el desarrollo de un programa haciendo uso de enclaves seguros.

En este punto se desglosará la finalidad principal del trabajo en varios objetivos parciales para poder abarcar la solución de este objetivo de forma estructurada y de una manera más sencilla.

Uno de los objetivos parciales del presente trabajo es la realización del diseño de alto y bajo nivel para el despliegue del programa que se encarga de realizar una consulta DNS mediante el uso de los enclaves.

Otro de los objetivos parciales que tiene este trabajo es la realización de las pruebas al despliegue realizado para verificar que realmente se está dotando de mecanismos de seguridad al protocolo y determinar qué grado de seguridad se le ofrece al protocolo DNS.

3.2 Alcance

Este proyecto se divide en tres fases:

- La primera fase consiste en el estudio y análisis de las diferentes tecnologías y tipos de enclaves existentes hoy en día que ofrecen mecanismos de seguridad a protocolos de nivel de aplicación. Este análisis nos ha permitido conocer que solución ofrece un mayor grado de seguridad, así como el tipo de vulnerabilidades que son evitadas a la hora de aplicar dichos mecanismos.
- En la segunda fase se ha realizado una implementación de un código que simula una consulta DNS a un servidor haciendo uso del tipo del enclave más seguro seleccionado en la fase anterior.
- La tercera fase del proyecto ha sido la de validación de la solución implementada. Aquí se ha realizado la consulta mientras se realizan ataques de diferentes tipos para ver cómo protege la consulta DNS de sus vulnerabilidades.

4 BENEFICIOS

En este apartado se indicarán cuáles son los beneficios que el Trabajo de Fin de Máster puede aportar. Dichos beneficios se clasificarán en tres subgrupos diferentes: los beneficios económicos, los beneficios tecnológicos y los beneficios sociales.

4.1 Económicos

Desde el punto de vista económico, la solución que aporta este trabajo contribuye a reducir drásticamente los costos asociados con el mantenimiento y recuperación de sistemas afectados por ataques de "Man in the Middle" y otras vulnerabilidades. Por otra parte, al prevenir eficazmente estos ataques, las organizaciones podrían evitar pérdidas financieras relacionadas con el robo de información confidencial y el deterioro de la reputación empresarial.

Otro de los beneficios que tienen los enclaves seguros de cara al usuario es la posibilidad de prescindir de equipamiento hardware y licencias software externas, como los antivirus, para poder dotar al protocolo de seguridad. De esta forma, se pueden crear mecanismos de seguridad en el propio equipo gracias al Entorno de Ejecución Fiable que, ofrecen los enclaves. Lo que permite el ahorro económico del equipamiento y licencias software utilizadas para proteger el equipo.

4.2 Tecnológicos

Además, al proteger de manera más efectiva contra ataques de usurpación de identidad o hombre en medio (Man in the Middle), esta solución contribuiría a salvaguardar la integridad de datos críticos a nivel global, fortaleciendo la estabilidad y resiliencia de la infraestructura digital.

Respecto a los beneficios tecnológicos, el trabajo demuestra cómo la incorporación de enclaves seguros en el protocolo DNS ofrece una mayor seguridad en las comunicaciones a través de Internet. Estos enclaves permiten el aislamiento y la protección de las operaciones críticas del DNS, como la traducción del nombre de dominio, evitando que ataques maliciosos comprometan el funcionamiento del sistema. Al hacerlo, se lograría un nivel de seguridad mayor, ofreciendo una base sólida para el desarrollo de nuevas tecnologías y servicios digitales que requieran un alto grado de protección y confianza.

4.3 Sociales

La implementación del protocolo DNS a través de enclaves seguros proporciona una mayor protección de la seguridad y privacidad para los usuarios finales, lo que se traduce en una mayor confianza a la hora de utilizar las tecnologías digitales, lo que impulsará una participación más amplia en el ecosistema digital.

5 ESTADO DEL ARTE

Este apartado contiene el estado del arte en el uso de enclaves seguros para dotar al protocolo DNS de mecanismos de seguridad, ya que actualmente, dicho protocolo carece de mecanismos de seguridad.

Analizamos, primeramente, el protocolo DNSSEC o Extensión de Seguridad DNS (DNS Security Extension) para ver los mecanismos de seguridad que ofrece y las vulnerabilidades que tiene. Posteriormente, se analizan de forma exhaustiva los diferentes tipos de enclaves seguros existentes hoy en día para poder realizar una consulta DNS segura y dotando al protocolo de los mecanismos necesarios para suplir las vulnerabilidades de DNSSEC. A continuación, se muestran los diferentes tipos de tecnologías utilizados por diferentes proveedores de procesadores para desplegar los enclaves seguros:

- Extensión de seguridad (SGX Secure Guard Extension) de los procesadores Intel
- Tecnología de Encriptado Segura (AMD Memory Encryption Technology) de los procesadores AMD
- ARM trustZone de los procesadores ARM

5.1 DNSSEC

DNSSEC ofrece los servicios de resolución de nombres de dominio ofrecidos por DNS tradicional. Aunque, hace uso del mecanismo de firma digital y criptografía de clave pública para ofrecer autenticación a la consulta DNS, no se firman, ni la petición, ni la respuesta DNS, son los propios datos del propietario los que están firmados. Esto ofrece una autenticidad de la información recibida a los usuarios [3].

La utilización de mecanismos de criptografía pública o asimétrica, el cual se basa en el uso de un par de claves pública y privada asociadas para encriptar y desencriptar el mensaje, aumenta la seguridad ofrecida por el protocolo DNSSEC, solucionando algunas de las vulnerabilidades identificadas en su predecesora, es decir, en el protocolo DNS, que se basa en el envío de la información mediante el protocolo UDP, el cual manda la respuesta en texto plano.

5.1.1 Servicios de seguridad ofrecidos por DNSSEC

DNSSEC utiliza criptografía de clave pública para firmar digitalmente los datos DNS. Cada nombre de dominio tiene una clave pública que se utiliza para firmar los datos DNS, y una clave privada correspondiente que se utiliza para verificar la firma. Cuando un usuario solicita un registro DNS, se comprueba que la firma es válida y no ha sido manipulada.

DNSSEC ofrece servicios de autenticación de origen y garantiza la integridad de la información DNS enviada, incluyendo mecanismos de denegación de autenticidad la existencia de los datos. No obstante, DNSSEC no está diseñado para ofrecer confidencialidad ni protección frente a ataques de denegación de servicios, es decir, tampoco ofrece disponibilidad [5]. Dentro de DNSSEC hay diferentes registros DNS utilizados para ofrecer los mecanismos de seguridad. Estos son los registros que hay:

- DNS RRsets son un grupo de registros DNS relacionados que comparten el mismo nombre de dominio y tipo.
- RRSIG es un tipo de registro DNS que proporciona firmas digitales a otros tipos de registros DNS.
- DNSKEY RR es un tipo de registro DNS utilizado para almacenar las claves públicas utilizadas para un nombre de dominio.
- NSEC es el registro DNS encargado de registrar una respuesta DNS que devuelve un nombre de dominio no autenticado.
- Signs zone es la zona cuyos RRsets están firmados y que contienen los registros DNSKEY, RRSIG correctamente construidos y Next Secure (NSEC).

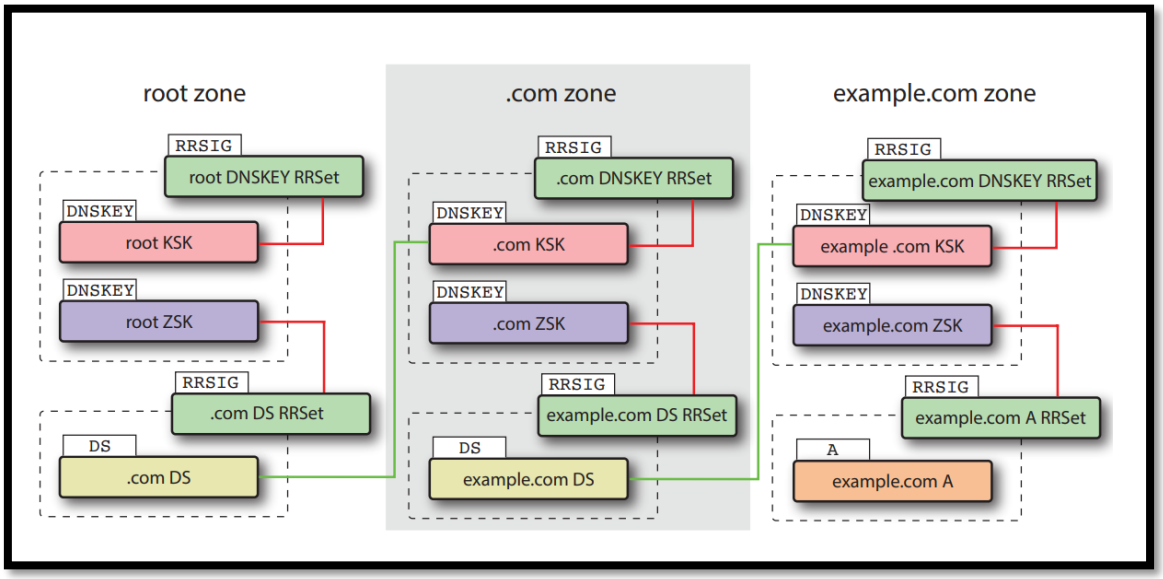


Figura 1: Perspectiva general de una consulta mediante DNSSEC [16]

5.1.1.1 Autenticación del origen e Integridad de la información

DNSSEC proporciona autenticación mediante la asociación de firmas digitales generadas criptográficamente con DNS RRsets, las cuales son almacenadas en un nuevo registro de recursos, el registro RRSIG. Es importante subrayar que la clave que firma la información de una zona está asociada a esta misma zona y no con los servidores de nombre autoritativos ubicados en dicha zona [4].

La autenticación del origen se realiza cuando un resolutor de nombres seguro (Security-aware resolvers) aprende la clave pública de una zona mediante la configuración de un anclaje de confianza (trust anchor) en el propio resolutor de nombres o mediante una resolución normal de DNS. Para hacer uso de esta última, las claves públicas de las zonas deben estar almacenadas el registro DNSKEY RR y, además, la propia clave pública del destino debe estar firmada por una clave de autenticación configurada o por otra clave que haya sido autenticada previamente [4].

Los resolutores de nombres seguros autentican la información de zona formando una cadena de autenticación formada desde la clave pública más reciente hasta la clave pública previa ya autenticada ya configurada en el resolutor. Por lo que se puede concluir que el resolutor debe tener configurado al menos un anclaje de confianza [4].

Para que los resolutores de nombres puedan realizar la cadena de autenticación, los servidores resolutores de nombres enviarán las claves necesarias para autenticar la clave pública de una zona en un mensaje DNS replay junto con su clave pública [4].

5.1.1.2 Autenticación de nombre y tipo de Inexistencia

La autenticación de origen solo permite firmar RRsets existentes de una zona. No obstante, DNSSEC hace uso de NSEC que permite al resolutor de nombres seguro autenticar una respuesta DNS negativa por inexistencia de nombre o tipo con los mecanismos de autenticación utilizados para autenticar respuestas DNS del caso anterior. Las cadenas de registros generadas con NSEC describen explícitamente los huecos, o "espacios vacíos", entre los nombres de dominio de una zona y enumeran los tipos de RR presentes en los nombres existentes [4].

5.1.2 Vulnerabilidades de DNS solucionadas por DNSSEC

Existen varias clases distintas de amenazas para el DNS, la mayoría de las cuales son instancias relacionadas con el DNS de problemas más generales, pero unas pocas son específicas de peculiaridades del protocolo DNS [5]. A continuación, se muestra cuáles son las amenazas solucionadas por DNSSEC.

5.1.2.1 Intercepción de paquetes (Packet Interception)

Los ataques de intercepción de paquetes es una de las formas más comunes que se realizan al protocolo DNS. Existen diferentes formas de intercepción de paquetes como pueden ser man-in-the-middle, interceptación de peticiones combinadas con respuestas falsas que se adelantan a la respuesta real de vuelta al resolver [5]…

En cualquiera de estos escenarios, el atacante hace creer o decir a cualquiera de las dos partes, normalmente al resolutor de nombres, lo que quiere que esa parte quiera para obtener la información almacenada en los mensajes transmitidos. Debido a que DNS va típicamente encapsulado en un datagrama UDP del nivel de transporte, los paquetes DNS son especialmente vulnerables y fáciles de interceptar, ya que no están encriptados.

No obstante, como se ha visto previamente, DNSSEC ofrece mecanismos de integridad de la información extremo a extremo. Por lo que ofrece una protección bastante efectiva frente a este tipo de ataques.

5.1.2.2 Predicción de consultas e Identidades (ID Guessing and Query Prediction)

Como una consulta DNS es transportado a través de UDP/IP, este tipo de ataque se basa en la generación de paquetes que coincidan con los parámetros del protocolo de transporte. Esto es debido a que el campo ID de la cabecera DNS tiene una longitud de 16 bits y el puerto UDP del servidor asociado a DNS es un “well-known”, por lo que tiene una cantidad máxima de 232 combinaciones posibles para encontrar la ID y el puerto UDP del cliente para la comunicación entre cliente y servidor. Esto no es un número muy largo, lo que hace que no sean medidas seguras frente a un ataque de fuerza bruta. Además, en la práctica, tanto el puerto UDP como

el ID del cliente puede ser precedido por el tráfico intercambiado previamente entre el cliente y el servidor. Lo que hace que se reduzca la cantidad de combinaciones posibles a 2^{16} [5].

Al igual que los ataques de intercepción de paquetes, al utilizar DNSSEC, un resolutor de nombres que compruebe las firmas utilizadas puede detectar las respuestas falsificadas y proteger el cliente de este tipo de ataques [5].

5.1.2.3 Encadenamiento de nombre (Name chaining)

Las amenazas de encadenamiento de nombres son amenazas específicas de DNS. Este tipo de ataques se basan en la modificación del mensaje DNS de respuesta de forma que el atacante pueda introducir nombres DNS arbitrarios a su elección y añadir información adicional relacionada con dichos nombres [5]. La forma general de un ataque de encaminamiento de nombres se guía por los siguientes pasos:

1. La víctima realiza una consulta DNS a la instancia del atacante o a una instancia de un tercero. En algunos casos, la consulta en sí no tiene por qué estar relacionada con el nombre atacado, es decir, el atacante utiliza la consulta de la víctima únicamente como medio para añadir información falsa acerca de otro nombre.

2. El atacante añade la respuesta, ya sea a través de intercepción de paquetes o a través de un servidor de nombres legítimo que está involucrado en el proceso de respuesta a la consulta realizada por la víctima.

3. La respuesta del atacante incluye uno o más RRs con nombres DNS en el campo RDATA (como de RR que hace referencia a la información asociada para proporcionar información acerca del nombre de dominio) con el objetivo de redirigir la siguiente etapa de la consulta al servidor elegido por el atacante, o para añadir información falsa de los nombres de dominio solicitados en el cache en la sección adicional de la respuesta.

DNSSEC ofrece mecanismos de defensa frente a la mayoría de variaciones de esta clase de ataques. Esto se puede realizar mediante la verificación de firmas, un resolutor de nombres puede determinar si los datos asociados a un nombre fueron realmente añadidos por una autoridad delegada para esa sección del espacio de nombres DNS.

5.1.2.4 Falso servidor de confianza (Betrayal By Trusted Server)

Este tipo de ataques es una variación del ataque de intercepción de paquetes. Sin embargo, desde el punto de vista del protocolo DNS, la única diferencia entre estos dos ataques es que, en el caso del falso servidor de confianza, el cliente envía la petición DNS de forma voluntaria al atacante [5].

Para ofrece protección frente a este tipo de ataques, basta con que el resolutor de nombres verifique las firmas utilizadas o haga uso del TSIG (mecanismo de seguridad para autenticación de mensajes DNS a través de dos servidores DNS) para autenticar los servidores confiables elegidos [5].

5.1.3 Puntos débiles de DNSSEC

En este subapartado se va a realizar un análisis de los puntos débiles o problemas que tiene la implementación del protocolo DNSSEC frente al protocolo tradicional DNS.

- El primer problema que tiene es su compleja implementación en las fronteras de diferentes zonas y la corrección de errores que pueden surgir debido a la configuración de zonas o claves caducadas.
- Otra de las desventajas frente a DNS es el incremento significativo que tienen los paquetes de respuesta, lo que hace que los servidores DNS compatibles con DNSSEC sean más vulnerables a ataques de negación de servicios.
- Debido a la necesidad de validación de la respuesta DNSSEC, esto hace que la carga de trabajo que tengan los resolutores de nombres se incremente considerablemente, ya que este deberá realizar las validaciones de firmas y, en algunos casos, tendrá que realizar más consultas.

5.2 SGX Enclaves:

Un enclave es una región de memoria protegida que proporciona confidencialidad de la información y de las áreas de ejecución de código. Es una instancia de un Ambiente de Ejecución Fiable (Trusted Execution Enviroment TEE) que está asegurado por hardware [6]. Los objetivos de seguridad de SGX se pueden resumir a los siguientes puntos [7]:

- SGX debe garantizar la posibilidad de detectar cualquier violación de la integridad del código e información de una instancia del enclave, impidiendo cualquier acceso a la código e información modificados.
- SGX debe proteger la confidencialidad del código e información críticos de una instancia del enclave.
- SGX debe ofrecer soporte múltiple para la coexistencia de varias instancias de enclaves simultáneas y ofrecer una garantía de aislamiento entre dos instancias de enclaves diferentes.

Con la intención de cumplir con los objetivos Intel desarrolló Intel SGX para proteger el equipo contra ataques hardware como ataques software. A continuación, se muestran las propiedades que tiene Intel SGX para ofrecer protección tanto a nivel de software como a nivel de hardware [8].

- Protección a nivel de Hardware:
  - La memoria a la que acceden los enclaves está encriptada mediante el uso de algoritmos de encriptación estándar del sector (industry-standard). La firma de un enclave caracteriza el contenido y el diseño del enclave en el momento de la compilación. Si el contenido y el diseño del enclave no son correctos según la firma, el enclave no se inicializa y no se ejecuta. Si un enclave se inicializa, debe ser idéntico al enclave original y no se modifica en tiempo de ejecución.
  - En caso de acceder a la memora o conectar los módulos de la DRAM (RAM Dinámica) a otro sistema, solo proporciona acceso a la información cifrada.
  - La clave utilizada para encriptar la memoria cambia de forma aleatoria cada ciclo de encendido (por ejemplo, arranque, suspender o hibernar). La clave generada se almacena en la CPU del equipo y no es accesible.

- Intel SGX no protege el hardware del equipo frente a ataques de canal lateral o ingeniería inversa. No obstante, se pueden desarrollar los enclaves con las características necesarias para crear la protección frente a estos ataques.

- Protección a nivel de Software
  - La memoria del enclave no se puede leer o escribir desde el exterior del enclave independientemente del nivel de privilegios y modo de CPU que se esté utilizando. En caso de intentar acceder a la memoria, se devuelve la página de operación abortada.
  - No está permitido el acceso al entorno del enclave mediante la llamada de funciones clásicas como saltos, manipulación de registros o manipulación de pilas (stack). Sólo se puede realizar una llamada a las funciones de los enclaves a través de nuevas instrucciones que realizan varios controles de protección. Las funciones clásicas de llamadas se inicializan en el interior del enclave a través de código de enclave.

5.2.1 Estructuración de un proceso de SGX

Tal y como se muestra en la Figura 2, una aplicación haciendo uso de enclaves de Intel SGX está dividida en dos particiones. Por un lado, está el host que es un componente no seguro y, por otro lado, está el enclave que es el componente seguro.

La aplicación ubicada en la parte del host se encarga de inicializar el enclave y almacenar el código de la aplicación, los datos cacheados de la misma y la memoria se ejecuta dentro del enclave de forma que se protege la información sensible, siendo inaccesible para el host acceder a dicho código. Cuando el enclave devuelve el resultado de la operación al proceso ubicado en la parte del host, el contenido del enclave sigue estando en el espacio de memoria protegido [9].

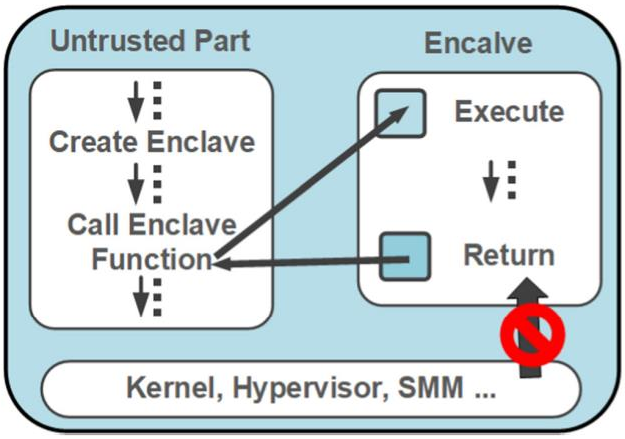


Figura 2: Proceso de ejecución de un enclave [7]

5.2.2 Estructuración de memoria en SGX

En un proceso SGX, como se ha comentado previamente, el código y la información crítica están almacenadas en un espacio de memoria aislado denominado Memoria Reservada del Procesador (PRM Processor Reserved Memory). La memoria PRM es un subconjunto interno de la Memoria DRAM y el tamaño que abarca está definido la BIOS del equipo [7].

El rango de memoria PRM está dividido en dos partes: La Página de Caché de Enclaves (EPC Enclave Page Cache), que es un subconjunto de la memoria PRM y el árbol de integridad, que es el responsable de mantener la etiqueta el Código de Autenticación de Mensaje (MAC Message Authentication Code) de la información del EPC.

Como se ha comentado en los subapartados anteriores, SGX está diseñado para gestionar múltiples enclaves de forma simultánea. Por esa razón los EPCs están separados en páginas de 4KB, de forma que cada una de estas páginas pueda ser adjudicada a diferentes instancias de enclaves.

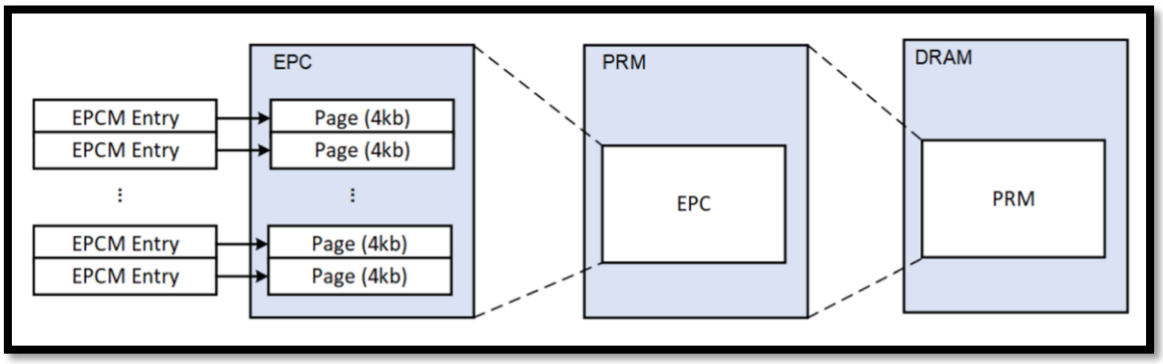


Figura 3: Organización de memoria SGX [7]

Para proteger la memoria EPC, SGX hace uso del Motor de Cifrado de Memoria (MEE Memory Encryption Engine) que es una extensión de Controlador de memoria integrado (IMC Integrated Memory Control) que tiene como finalidad él control de acceso al enclave, gestionando las llamadas de lectura y escritura a memoria. De esta forma, si se recibe una petición de acceso al EPC ajeno al enclave, esta será denegada por el MME, independientemente de que la petición sea solicitada por un sistema con privilegios [7].

Dado que el software responsable utilizado para la asignación de páginas EPC no es de total confianza, SGX almacena la información de asignación de páginas en un espacio de memoria denominado Mapa para Página de Chache de Enclave (EPCM Enclave Page Cache Map) que es un array en el que cada elemento corresponde con una página EPC [7].

5.2.3 Ciclo de vida de un enclave

Intel SGX almacena la información de cada enclave en áreas llamadas SGX Control de Estructuras de Enclaves (SECS SGX Enclave Control Structure) y cada SECS está almacenada en una página EPC dedicada. Esta área tiene asociada una dirección virtual tiene la finalidad de identificar el enclave cuando se invoque una instrucción SGX [7]. La Tabla 2 muestra las instrucciones SGX utilizadas para manipulación de enclaves:

Instrucción	Descripción
ECREATE	Declarar la base y el rango, creación del enclave
EADD	Añade un página de 4KB
EEXTEND	Medición de 256 bytes
EINT	Declarar el enclave

Instrucción	Descripción
EREMOVE	Eliminar página
EENTER	Acceder al enclave
ERESUME	Información del Resumen del enclave
EEXIT	Salir del enclave
AEX	Salida asíncrona del enclave
ECALL	Mandar una instrucción al enclave para que una aplicación ajena al enclave invoque funciones internas

Tabla 2: Descripción de instrucciones SGX [8]

El ciclo de vida de un enclave está separado en cuatro fases: La fase de creación, la fase de carga, la fase de inicialización y la fase de liberación [7]:

4. La fase de creación consiste en cargar un SECS particular de un nuevo enclave en una página de la memoria EPC mediante la instrucción ECREATE de la Tabla 2.

5. La fase de carga consiste en cargar el código e información inicial dentro del enclave creado en la fase de creación mediante la instrucción EADD. Mientras tanto, el software del sistema utiliza la instrucción EEXTENDED para actualizar la medición del nuevo enclave creado.

6. En la fase de inicialización, el software utiliza la instrucción EINT para lanzar el enclave creado y el atributo INIT del enclave pasará a estar a verdadero (true). Después, una aplicación podrá correr el código cargado en el enclave. Una vez que el enclave entra en esta fase, no se puede volver a llamar a la instrucción EADD.

7. La fase de liberación empieza cuando el enclave finaliza la ejecución del código interno que tenía y se llama a la instrucción EREMOVE. Esto generará una redistribución de las páginas EPC utilizadas por el enclave. Un enclave es liberado cuando una página particular EPC que tiene un SECS determinado es liberada.

5.3 AMD Memory Encryption Technology

Al igual que los procesadores Intel utilizan la extensión SGX, los procesadores AMD utilizan la Tecnología de Encriptado de Memoria (MET Memory Encryption Technology), que tiene como objetivo ofrecer cifrado y protección a la memoria del sistema mediante TEE asistido por hardware [10].

La Tecnología de Encriptado de Memoria AMD introduce mecanismos de encriptación AES 128 dentro del Sistema en Chip (System on Chip SoC) que encripta y desencripta la información de forma transparente cuando dicha información entra o abandona el SoC respectivamente. En base a la Tecnología de Encriptado de Memoria, ADM propone dos funciones de seguridad principales denominadas Encriptado Seguro de Memoria (Secure Memory Encryption SME) y Encriptado Seguro de Virtualización (Secure Encrypted Virtualization SEV), ambos son gestionados por el sistema operativo o por el hipervisor (hypervisor) y no son necesarios ningún cambio en las aplicaciones softwares [10].

Respecto a las claves de encriptación utilizadas en ADM, su gestión, generación, almacenamiento y suministro se llevan a cabo a través del proceso seguro de AMD. Además, dichas claves se mantienen ocultas de la parte no segura de la plataforma [10].

5.3.1 Secure Memory Encryption (SME)

El SME es una característica de seguridad que ayuda a proteger el contenido de la memoria del sistema cifrándolo en tiempo real. Esta función está diseñada para evitar el acceso no autorizado a datos confidenciales almacenados en la memoria [11].

La encriptación de memoria se realiza mediante hardware dedicado en los controladores de memoria. Cada controlador ofrece un alto rendimiento del motor Estándar Avanzado de Encriptado (Advanced Encryption Standard AES) que encripta la información cuando es escrita en la DRAM y la desencripta cuando se lee la información de la memoria, tal y como se muestra en la siguiente Figura 4.

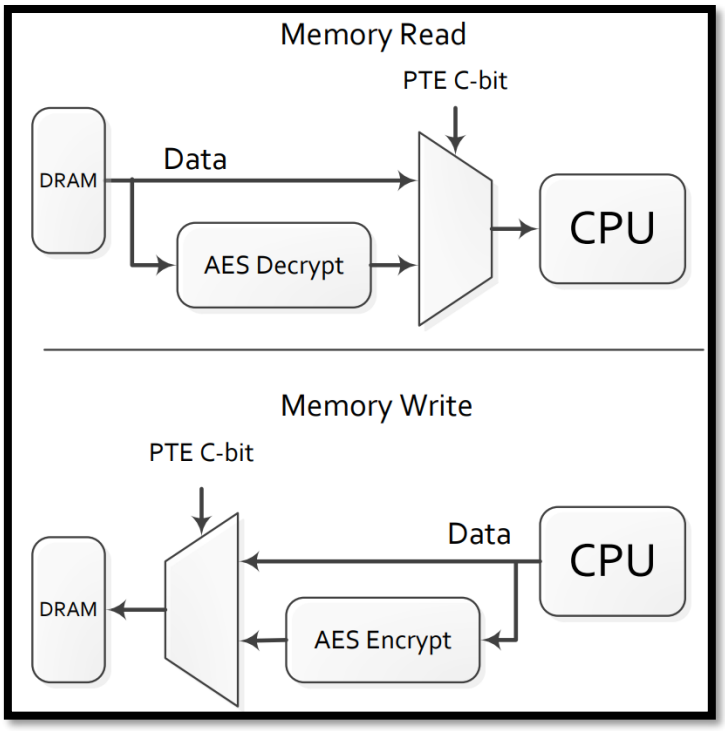


Figura 4: Comportamiento del Encriptado de Memoria [11]

El encriptado de la información se realiza con una clave de 128 bits un modo que utiliza un ajuste adicional basado en una dirección física adicional para proteger contra los ataques de movimiento de bloques de texto cifrado. Esta clave se genera de forma aleatoriamente cada vez que se resetea el equipo y no es visible para ningún software ejecutado en el núcleo de la CPU. La clave es gestionada íntegramente por el Procesador Seguro AMD (AMD-SP), el cual es un microcontrolador que funciona como un subsistema de seguridad dedicado que va integrado en AMD SOC [11].

El sistema operativo (SO) o el Hipervisor (HV), mediante el software de gestión de tablas de páginas, se encargan de realizar el control de las páginas de la memoria que van a ser encriptadas. Una vez que se habilite la opción de encriptado de memoria, se utiliza el bit 47 de la dirección física para indicar si la dirección de memoria está protegida o no, tal y como se muestra en la Figura 5. Esto pasa cuando el SO o el HV ponen dicho bit a 1 en la entrada de tablas de página (Page Table Entry PTE), lo que genera que se realice el encriptado o desencriptado previo al acceso a dicha página de memoria mediante el motor AES del controlador de memoria [11].

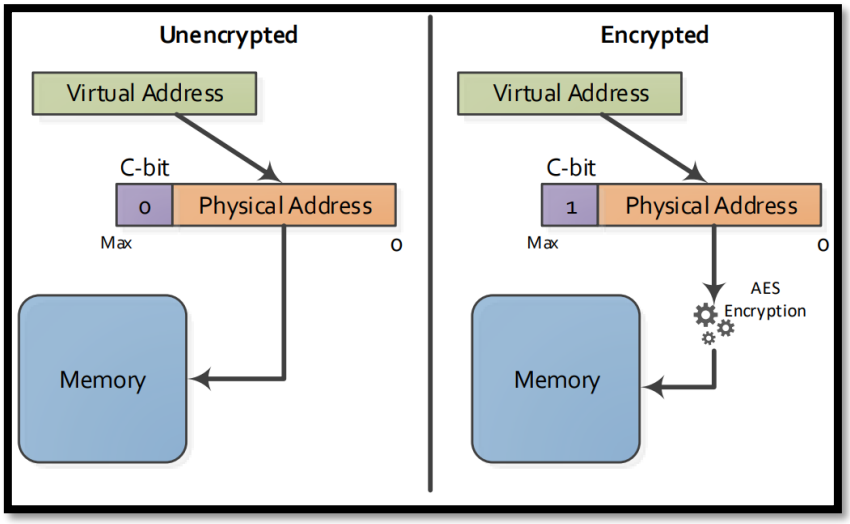


Figura 5: Mapeo de memoria para encriptar una página de memoria [11]

5.3.1.1 Encriptado de Memoria Completa

Existen dos modelos de uso de SME. En primer lugar, está el encriptado de memoria completa. Con este modo, se encripta todo el contenido de la memoria DRAM mediante una clave generada automáticamente. Esto ofrece una protección alta frente ataque contra inicio lento (cold boot), intrusión de interfaz de DRAM (DRAM interface snooping) y ataques similares [11].

Como se ha comentado previamente, el SO y el HV se encargan de encriptar la información de a almacenar en la memoria. En este caso, para encriptar toda la memoria, el SO o el HV ponen el C-bit o el bit 47 de todas las direcciones físicas a 1 en todas las tablas de páginas incluyendo las páginas de instrucción y las páginas de la propia tabla de páginas. En el caso de utilizar el HV, todas las máquinas virtuales controladas por HV usan la misma clave para encriptar todas las direcciones físicas [11].

5.3.1.2 Encriptado de Memoria Parcial

El método de uso del bit 47 en las tablas de páginas permite proporcionar al SO y HV una flexibilidad para seleccionar únicamente el subgrupo de la memoria que se desea encriptar. Con esto, aparte de ofrecer protección de encriptado de memoria, también ofrece una mejora en el rendimiento para el almacenamiento de la información no sensible. Este tipo de usos se utiliza para ofrecer una capa de aislamiento para un volumen de trabajo crítico, de forma que se encripte únicamente la zona de la memoria donde se ubica la información sensible y agilizar los

procesos de lectura y escritura para la zona de memoria donde se almacena la información no sensible [11].

5.3.2 Secure Encrypted Virtualization (SEV)

El Encriptado Seguro Virtualizado es una característica añadida la arquitectura de AMD y es una tecnología x86 diseñada para aislar máquinas virtuales del hipervisor. Esto permite que a cada máquina virtual individual se le asigne una clave de encriptado AES única utilizada para encriptar automáticamente la información en uso de dicha máquina virtual. Cuando el hipervisor se disponga a leer la memoria en el interior de un invitado (guest), solo será capaz de visualizar los bytes encriptados [12].

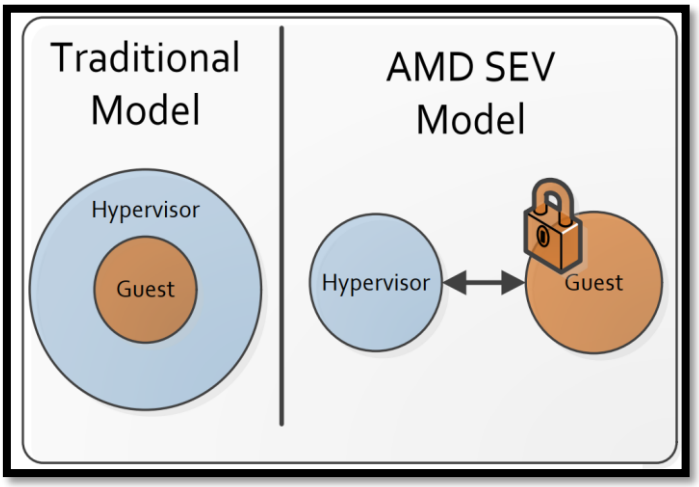


Figura 6:Modelo de seguridad SEV [11]

La documentación ofrecida por los desarrolladores del procesador AMD [12] detallan que actualmente, está en uso la nueva generación SEV o SEV-SNP o SEV-Paginación Anidada Segura (SEV-Secure Nested Paging). SEV-SNP ofrece la funcionalidad de cifrar la información de cada máquina virtual y, además, ofrece una fuerte protección de la integridad para prevenir ataques basados en hipervisor como reproducción en la información o remapeo de memoria.

Respecto al modelo de detalle de amenazas representado en la Figura 7, SEV-SNP detecta todos los componentes software de la CPU y los dispositivos PCI como entornos no seguros, al igual que la BIOS del sistema anfitrión, el hipervisor, otras máquinas virtuales…

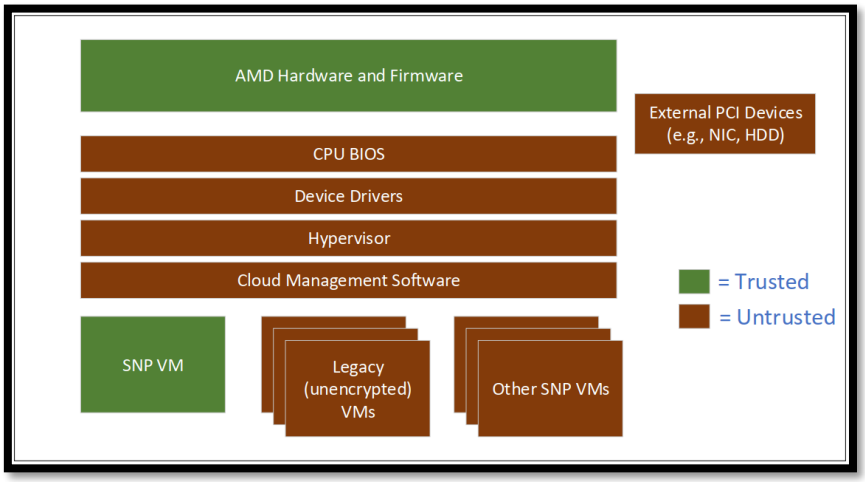


Figura 7: Modelo de amenazas de SEV-SNP [12]

SEV-SNP utiliza el modelo de amenaza de hipervisor "benigno pero vulnerable", que se define como que el hipervisor no es fiable al 100% pero que se confía de que el hipervisor actúa con intenciones lícitas. Con este modelo, SEV-SNP dota las máquinas virtuales con mecanismos de seguridad que comprometan los siguientes aspectos [12]:

- **Confidencialidad**: Los ataques que amenazan la confidencialidad se solucionan mediante el encriptado de memoria basado en hardware visto en apartados anteriores. Esto hace que los componentes no fiables como el hipervisor o el dispositivo DMA sean incapaces de leer el texto plano almacenado dentro de la máquina virtual. Además, SEV-SNP añade protección de confidencialidad para el estado de registro de la máquina virtual, encriptando este estado cuando el hipervisor vuelve a tener el control.
- **Integridad**: SEV-SNP está diseñado para prevenir ataques basados en la integridad de software y reducir riesgos asociados al comprometimiento de la integridad de memoria. El principio de integridad de SEV-SNP funciona de forma que, si una máquina virtual es capaz de leer un dato encriptado de una página de memoria, siempre tiene que ser capaz de leer el último dato escrito. También se asegura la integridad cuando la página de memoria de una máquina virtual se cambia al disco o si la máquina virtual se cambia a un nuevo host. La forma de asegurar esta integridad es mediante la comunicación del hardware y firmware de la CPU.
- **Disponibilidad**: Existen dos tipos de ataques basados en la denegación de servicios que pueden comprometer la disponibilidad de la máquina virtual. Por un lado, que la máquina virtual deniegue el servicio al hipervisor y, por otro lado, que un hipervisor malicioso no permita la ejecución de una máquina virtual. La tecnología SEV solo ofrece protección frente al primer caso, haciendo que la máquina virtual invitada sea incapaz de inutilizar el hipervisor o hacer que la máquina física quede inutilizable. Esto se realiza mediante una interrupción física de temporizador, que hace que el hipervisor obtenga el control cuando él quiera.
- **Acceso físico**: Para los ataques de acceso físico, la tecnología SEV-SNP ofrece seguridad frente a ataques de DRAM "cold boot" que se basan en obtener acceso no autorizado a la información sensible almacenada en la memoria de un ordenador en el periodo de

enfriamiento (cool down) para extraer el contenido una vez desconectada la alimentación del equipo. Esto se realiza mediante el encriptado la memoria o subgrupo de ella en la que se almacena la información sensible.

En la Figura 8 se muestra un resumen de las posibles amenazas que pueden explotar las vulnerabilidades y comprometer la seguridad de una máquina virtual y si la tecnología de Encriptado Seguro de Virtualización y sus actualizaciones ofrece protección frente a estos ataques:

✓ = Mitigated ★ = Optionally Mitigated ⊘ = Not Mitigated	SEV	SEV-ES	SEV-SNP
Potential Threats			
Confidentiality			
VM Memory *Example attack: Hypervisor reads private VM memory*	✓	✓	✓
VM Register State *Example attack: Read VM register state after VMEXIT*	⊘	✓	✓
DMA Protection *Example attack: Device attempts to read VM memory*	✓	✓	✓
Integrity			
Replay Protection *Example attack: Replace VM memory with an old copy*	⊘	⊘	✓
Data Corruption *Example attack: Replace VM memory with junk data*	⊘	⊘	✓
Memory Aliasing *Example attack: Map two guest pages to same DRAM page*	⊘	⊘	✓
Memory Re-Mapping *Example attack: Switch DRAM page mapped to a guest page*	⊘	⊘	✓
Availability			
Denial of Service on Hypervisor *Example attack: Malicious guest refuses to yield/exit*	✓	✓	✓
Denial of Service on Guest *Example attack: Malicious hypervisor refuses to run guest*	⊘	⊘	⊘
Physical Access Attacks			
Offline DRAM analysis *Example attack: Cold boot*	✓	✓	✓
Active DRAM corruption *Example attack: Manipulate DDR bus while VM is running*	⊘	⊘	⊘
Misc.			
TCB Rollback *Example attack: Revert AMD-SP firmware to old version*	⊘	⊘	✓
Malicious Interrupt/Exception Injection *Example attack: Inject interrupt while RFLAGS.IF=0*	⊘	⊘	★
Indirect Branch Predictor Poisoning *Example attack: Poison BTB from hypervisor*	⊘	⊘	★
Secure Hardware Debug Registers *Example attack: Change breakpoints during debug*	⊘	⊘	★
Trusted CPUID Information *Example attack: Hypervisors lies about platform capabilities*	⊘	⊘	★
Architectural Side Channels *Example attack: PRIME+PROBE to track VM accesses*	⊘	⊘	⊘
Page-level Side Channels *Example attack: Track VM access patterns through page tables*	⊘	⊘	⊘
Performance Counter Tracking *Example attack: Fingerprint VM apps by performance data*	⊘	⊘	⊘

Figura 8: Ataques que protege SEV y sus actualizaciones [12]

5.4 ARM TrustZone

TrustZone hace referencia al nombre de la arquitectura de seguridad que utilizan los procesadores ARM y ofrece dos ambientes de ejecución con aislamiento forzado en todo el sistema entre ambos, tal y como se muestra en la Figura 9 [13]:

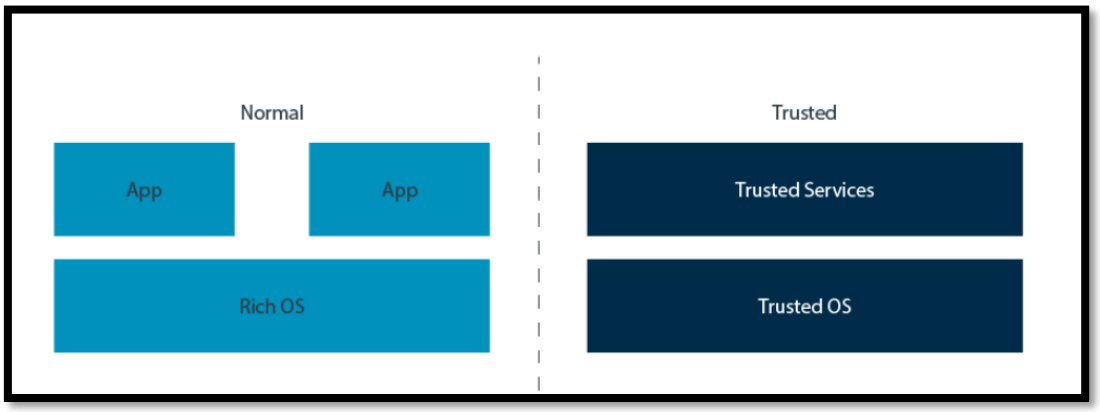


Figura 9: mundo normal y confiable o seguro de los procesadores AMR [13]

Como se puede observar en la Figura 9, por un lado, el mundo normal se ejecuta en una pila de software largos y complejos. Debido al tamaño de la pila de software, hace que sea más vulnerable a sufrir ataques. Por otro lado, el mundo confiable o seguro se ejecuta en un Entorno de Ejecución Seguro, por lo que el núcleo en el que se ejecuta es más liviano, por lo que la pila software del mundo confiable es más simple y pequeño, es decir, que reduce las vulnerabilidades y riesgos de ataques [13].

Al igual que la extensión SGX de Intel, en la arquitectura de los procesadores AMR, existen dos estados de seguridad: El estado seguro y el estado no seguro, esto se puede ver en la Figura 12 [14].

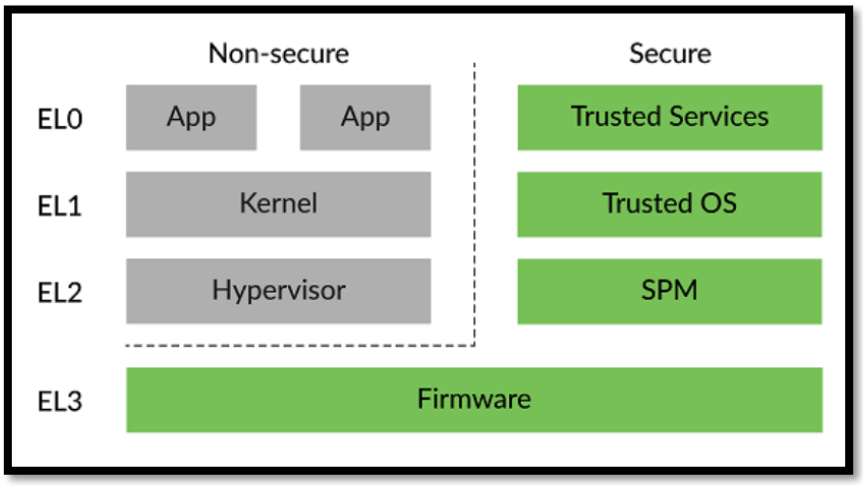


Figura 10: Estados de seguridad en la arquitectura ARM [14]

El estado de seguridad define los niveles de excepción se pueden llamar, las áreas de memoria a las que se pueden acceder y como están representados dichos accesos de memoria al sistema de bus de memoria. Por un lado, en el estado seguro, el elemento procesador (PE Processing Element) puede acceder al espacio de direcciones físicas del estado seguro y del estado no seguro, también puede acceder a la copia del banco de registros del estado no seguro. Por otro lado, en el estado no seguro que hace referencia al mundo normal visto en la Figura 9, el PE solo puede acceder al espacio de direcciones físicas no seguras. Además, solo tiene acceso al sistema de registros no seguro [15].

La arquitectura ARM permite esta la división mediante la implementación de diferentes niveles de servicio. Esos niveles de servicio se denominan niveles de excepción o EL abreviado. Los niveles de excepción están numerados del 0 al 3 y el nivel de mayores privilegios tiene asociado el número más alto y estos se indican de la siguiente manera: El <x>, donde x es el número de excepción. Los niveles excepción cambian cuando ocurre uno de los siguientes eventos:

- Cuando el procesado genera una excepción
- Cuando se regresa de una excepción
- Reiniciado del proceso
- Durante el estado de depuración
- Al salir del estado de depuración.

La siguiente imagen muestra los niveles de excepción existentes en la arquitectura ARM [15]:

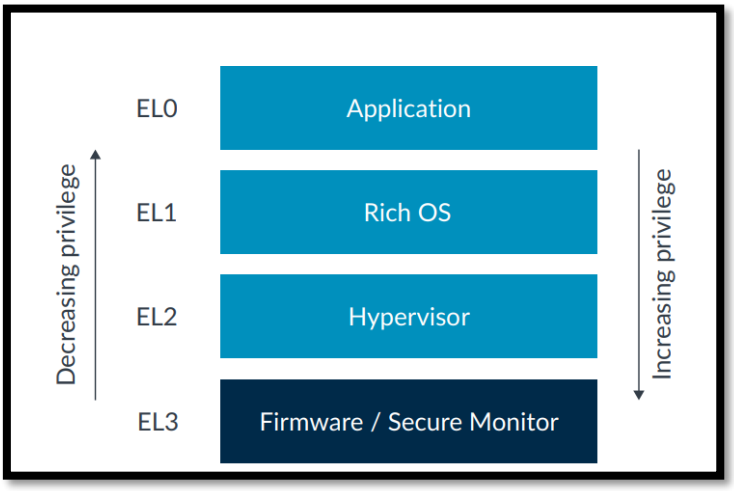


Figura 11: Niveles de Excepción en la arquitectura ARM [15]

5.4.1 RME Realm Management Extension

Con la intención de ofrecer mayor seguridad y aislamiento de la información, los procesadores ARM incorpora la Extensión de Gestión Realm (RME Realm Management Extension), la cual es el componente hardware de la Arquitectura de Computación Confidencial ARM (ARM CCA ARM Confidential Compute Architecture) que también incluye elementos software. La finalidad de RME es la transferencia dinámica de recursos y memoria a un espacio de direcciones protegida e inaccesible para softwares con privilegios o para el firmware de TruistZone [14].

La incorporación de la RME genera un nuevo estado de seguridad denominado Realm, que se define como un entorno de ejecución protegido que lo genera la ARM CCA, esto se puede ver en la Figura 12 [14]. Además, los Realms tienen la posibilidad de proteger la información de los softwares con bajos privilegios como las máquinas virtuales. Los Relams previenen la ejecución de ataques mediante software que se ejecuta con niveles de privilegio superiores, como un SO o un hipervisor. A pesar de que un software con altos privilegios es responsable de asignar y gestionar los recursos que utiliza un Realm, este no tiene acceso a la información que contiene el Realm ni afectar a los flujos de ejecución del mismo [14].

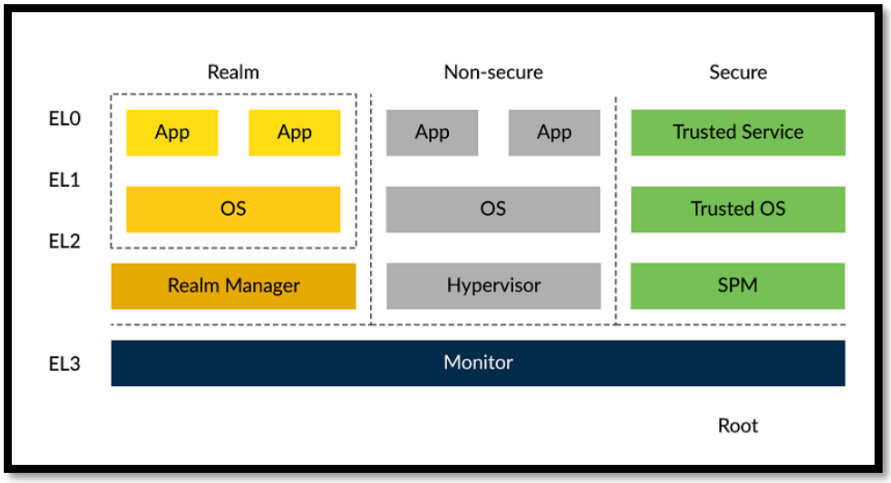


Figura 12: Estados de seguridad de los procesadores ARM [14]

Tal y como se puede observar en la Figura 12, la arquitectura de AMR ofrece un aislamiento entre siguientes estados:

- Aísla el estado seguro del estado no seguro y el estado realm.
- Aísla el estado realm del estado no seguro y el estado seguro.

Con RME, al ocurrir una excepción de nivel 3 se realiza el cambio de un estado de seguridad seguro a su propio estado seguro llamado root o administrador. De esta forma, RME es capaz de aislar las excepciones de nivel 3 de cualquier estado de seguridad, ya que el EL3 aloja la plataforma y el código de arranque inicial.

5.4.2 Control y gestión de los cambios de estados seguros

Mediante la modificación de los niveles de excepción, se puede pasar de un estado de seguridad a otro. Como se puede observar en la Figura 12, en los niveles EL0, EL1 y EL2, el procesador puede en los 3 estados de seguridad. El cambio de estado se realiza mediante la combinación de niveles de excepción y la modificación del registro de control SRC_Elx (Registro de configuración seguro), el cual se utiliza para controlar el estado de seguridad y el envío de excepciones a EL3 [14].

Dentro del registro SRC_EL3, los campos NS y NSE son los que controlan el cambio de estado de seguridad. La siguiente tabla muestra los valores de los bits NS y NSE y su respectivo cambio de estado [14]:

SCR_EL3.{NSE,NS}	Estado de Seguridad
0b00	Seguro
0b01	No Seguro
0b10	-
0b11	Realm

Tabla 3: Valores de bits NS y NSE con su estado de seguridad relacionado [14]

En el caso de pasar da un estado a otro, tal y como se muestra en la Figura 13, se sigue el siguiente procedimiento:

1. La ejecución empieza en el estado Realm estado el registro SRC_EL3.{NSE, NS} con valor 0b11. El software ejecuta la instrucción Monitor de Llamadas Seguro (SMC Secure Monitor Call) que genera una excepción pasando a EL3.
2. El procesador entra en el nivel de excepción 3 y entra en el estado de administrador. El software, estando en EL3, cambia los bits del registro SRC_EL3.{NSE, NS} al valor correspondiente para acceder al estado de seguridad deseado y ejecuta la instrucción de Devolver Excepción (ERET Excepction Reurn).
3. El ERET genera la salida del nivel de Excepción 3. Al salir del EL3, el registro SRC_EL3.{NSE, NS} controla el estado de seguridad al que se mueve.

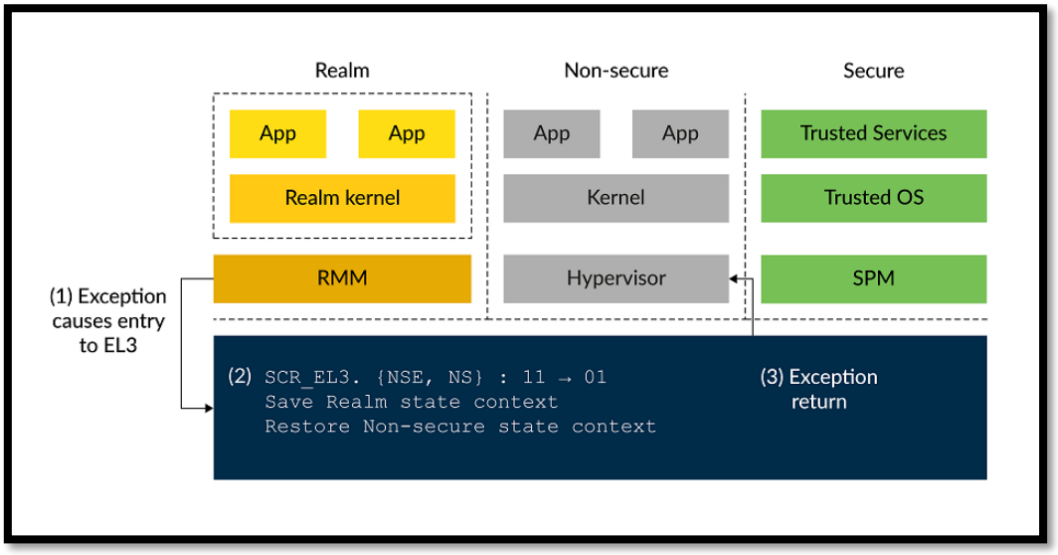


Figura 13: proceso de cambio de estado de seguridad [14]

Como solo hay una copia del vector de registros, de los registros de propósito general y de la mayoría de registros del sistema, al realizar el cambio de un estado de seguridad a otro, el software que solicita el cambio de estado se encarga de realizar la llamada al monitor seguro. El monitor seguro, una vez que se ha solicitado el cambio de estado, se encarga de guardar la información del estado no seguro en el registro de estado y recuperar la información almacenada en el registro del estado seguro [14].

5.4.3 Arquitectura de Computación Confidencial ARM

Como se ha comentado en puntos anteriores, el RME pertenece al ACC. En base a la documentación presentada por los procesadores ARM [16], la Computación Confidencial se define como la protección de información haciendo uso del rendimiento computacional en entorno seguro y fiable respaldado por hardware. Esto permite proteger la información y el código de observaciones o modificaciones de softwares privilegiados o agentes hardware. Cualquier aplicación o sistema operativo que se esté ejecutando en un entorno de computación confidencial que ofrece aislamiento frente a agentes no confiables en el resto del sistema. Cualquier información generada o consumida por este entorno aislado no puede ser observada por ningún otro programa o software en ejecución de la plataforma sin un permiso explícita. Para que el procesador permita el uso de ARM CCA, la plataforma debe proporcionar los siguientes aspectos:

- Un entorno de ejecución que ofrezca aislamiento frente a cualquier agente no fiable.
- Mecanismos para establecer un entorno de ejecución inicializada dentro del estado seguro o de confianza. Este establecimiento requiere que el entorno de ejecución tenga su propia Cadena de Confianza (Chain of Trust), independiente de la Cadena de Confianza que utilizan en paralelo los entornos de ejecución no fiables en la plataforma.

La arquitectura del sistema RME requiere que los Espacios de Direcciones Físicas (PAS Physical Address Space) de los estados de seguridad seguro, real y administrador estén encriptadas. Esto se realiza mediante la extensión de Contexto de Encriptado de Memoria (MEC Memory Encryption Contexts) ofrecido por RME. Los desarrolladores de los procesadores de ARM definen el MEC como una serie de configuraciones de encriptado que están asignadas a las áreas de memoria [16].

Para dotar a los softwares del sistema los mecanismos de encriptado, a estos se les asocia unos identificadores denominados MECID, que se definen como etiquetas identificadoras que están asociadas a con diferentes Contextos de Encriptado de Memoria. A los softwares del sistema se les asocian diferentes MECID. Esto se ve reflejado en Figura 14 donde a cada Realm se le asocia un MECID diferente dentro de un contexto de encriptado.

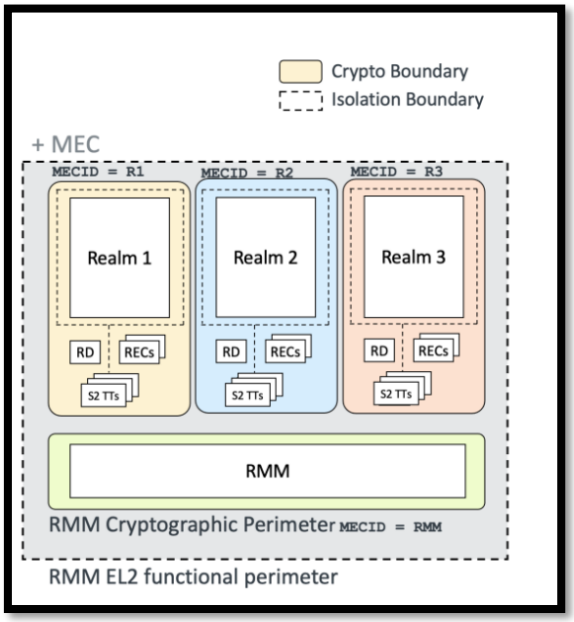


Figura 14: Esquema de MECDIs asociados a un MEC [16]

Los identificadores son asignados por software, para las PAS los diferentes Realms se realiza mediante el Monitor de Gestión Realm (RMM Realm Management Monitor), utilizando una combinación de valores de los registros del sistema y los bits de la tabla de páginas de memoria. MEC proporciona a los softwares, Relams y al RMM la opción de utilizar más de un registro MECID en conjunto con los bits de la tabla de páginas utilizado para seleccionar la región específica de memoria mapeada que hace referencia al registro MECID a dicho software. Esto permite al RMM usar un registro MECID para gestionar su propia estructura de información y otro MECID para gestionar los Realms que está gestionando [16].

Cuando se realiza una transacción en la que interviene un área de memoria asociada a un MECID, éste se utiliza para recuperar un tweak o clave de cifrado (un contexto de cifrado) que se utilizará para cifrar o descifrar la transacción. La clave de cifrado es un dato que modifica la operación de cifrado. Al arrancar el sistema, el motor de protección de memoria (MPE) genera y almacena un conjunto de contextos de cifrado. Estos contextos están indexados por el MECID y pueden actualizarse a partir de una petición del mundo raíz cuando un MECID está siendo reutilizado por una entidad diferente del sistema.

Las claves de encriptación MEC sólo deben almacenarse en registros de una sola escritura, accesibles únicamente mediante peticiones de raíz. Una vez que se restablece una clave, el valor debe establecerse en un valor predeterminado diferente de todas las demás claves de encriptación MEC activas.

5.5 Conclusiones

La conclusión principal que se puede sacar del estado del arte realizado es las diversas formas que existen de ofrecer seguridad al protocolo DNS. No obstante, las soluciones aplicables hoy

en día son bastante complejas de implementar y no ofrecen una seguridad integra o completa al protocolo.

La posibilidad y el amplio abanico de mecanismos de seguridad que ofrecen los Entornos de Ejecución seguros hacen que sea una solución más que atractiva para proteger protocolos que carecen de estos mecanismos de seguridad.

En el estudio realizado acerca de los mecanismos de atestación para Entornos de Ejecución Fiables [17] se realiza una comparación entre diferentes TEEs analizados en el presente proyecto y muestran los distintos mecanismos de seguridad que ofrecen tanto al software que se está ejecutando en estos entornos como a los programas que realizan el proceso de atestación.

Features	SGX	TrustZone	SEV			RISC-V			
			Vanilla	SEV-ES	SEV-SNP	Keystone	Sanctum	TIMBER-V	LIRA-V
Integrity	●	○	○	○	●	●	○	○	○
Freshness	●	○	○	○	●	●	○	○	○
Encryption	●	○	●	●	●	●	○	○	○
Unlimited domains	●	○	◑	●	●	●	●	●	○
Open source	◑	◑	○	○	○	●	●	●	○
Local attestation	●	○	○	○	○	○	●	●	○
Remote attestation	●	◑	●	●	●	●	●	●	●
API for attestation	●	◑	○	○	●	●	●	●	●
Mutual attestation	○	◑	○	○	○	○	○	○	●
User-mode support	●	●	●	●	●	●	●	●	○
Industrial TEE	●	●	●	●	●	○	○	○	○
Isolation and attestation granularity	Intra-address space	Secure world	Virtual machine	Virtual machine	Virtual machine	Secure world	Intra-address space	Intra-address space	Intra-address space
System support for isolation	Microcode	Secure monitor	Firmware	Firmware	Firmware	Secure monitor + PMP	Secure monitor + PMP	Tagged memory + MPU	PMP

Figura 15: Comparativa de mecanismos de seguridad entre Intel SGX y AMD SEV [17]

Si se observa la Figura 15, se aprecia es que los enclaves seguros de Intel SGX son los Entornos de Ejecución Fiables que ofrece un mayor grado de seguridad. Además, es el único entorno que permite realizar un proceso de atestación local. Esto indica que un programa externo al ejecutado en el enclave puede ser capaz de interactuar con el enclave y realizar llamadas al mismo para recibir o enviar información al programa interno del enclave.

Además, cabe destacar que los enclaves SGX son los únicos que ofrecen un entorno aislado, lo que hace que se ofrezca una mayor seguridad que el resto de entornos. No obstante, esto complica la tarea de atestar un programa externo al enclave.

6 ANÁLISIS DE ALTERNATIVAS

En este apartado se realiza un análisis de las diferentes tecnologías que se han descrito en el apartado de Estado del Arte. El análisis consistirá en realizar una comparación de dichas tecnologías para determinar cuál de ellas, de forma óptima, es la que ofrece un mayor grado de securización del protocolo DNS.

Además de comparar las diferentes tecnologías existentes para crear un Entorno de Ejecución Fiable, también se analizarán diferentes herramientas utilizadas para desplegar la solución que servirá para cumplir con el objetivo propuesto en el apartado 3 de este trabajo.

6.1 Descripción de alternativas

En este subapartado se realiza una descripción de las diferentes opciones existente para realizar el trabajo.

Además, Respecto a las herramientas a utilizar se basan en las diferentes librerías existentes para desarrollar las consultas DNS dentro de los Entornos de Ejecución Fiables y el equipamiento que se va a utilizar para poder desplegar la solución de forma óptima.

6.1.1 TEE y Enclaves seguros

Teniendo en cuenta lo que se ha descrito en el apartado anterior, existen diferentes tecnologías que proporcionan un Entornos de Ejecución Fiables para ofrecer seguridad a nivel de hardware. En este caso y para realizar la solución que cumpla con los objetivos del presente proyecto, se ha planteado el uso de las siguientes tres tecnologías:

- SGX enclaves de Intel
- TrustZone de ARM
- Tecnologías de Encriptación de Memoria de AMD.

La descripción y el funcionamiento de las tecnologías indicadas viene realizado de forma explícita en el punto 5 Estado del Arte del presente proyecto.

6.1.2 Despliegue de la solución

Una vez descritos las diferentes tecnologías para la creación de Entornos de Ejecución Fiables, el siguiente paso es la descripción del despliegue de la solución de este Trabajo de Fin de Máster.

La primera alternativa es **el despliegue del desarrollo basado en hardware,** es decir, desplegar la solución mediante un ordenador portátil personal que disponga del procesador necesario para crear el Entono de Ejecución Fiable. Esta solución implica prescindir del uso de librerías y softwares (como SDK) externos que simulen el uso de estos entornos. Además, una vez habilitada la parte hardware para estos entornos y realizada la configuración de los mismos, bastaría con empezar a programar la solución dentro de estas tecnologías para crear la solución del proyecto. No obstante, uno de los mayores inconvenientes que tiene esta alternativa es que los procesadores de los equipos tienen que ser compatibles con los entornos, es decir, que tiene que soportar e incluir la extensión de seguridad dentro del procesador para poder desplegar los

enclaves. Dado que estas tecnologías son relativamente recientes, pocos procesadores son los que ofrecen las extensiones necesarias y, además tienen un coste bastante elevado.

Otra alternativa que se plantea es la **virtualización completa de la solución a desarrollar**, es decir, utilizar emuladores capaces de virtualizar la parte hardware necesaria para realizar el despliegue de la solución. Uno de los emuladores capaces de realizar este proceso es QEMU [18] [19] el cual ofrece un modelo virtual de una máquina formada por CPU, memoria y dispositivos emulados) para ejecutar en un Sistema Operativo invitado y con la capacidad de soportar un gran número de hipervisores, denominados aceleradores. La siguiente tabla muestra una lista de las arquitecturas de CPUs que puede emular con sus respectivos aceleradores

Arquitectura CPU	Aceleradores
Arm	kvm (64 bit only), tcg, xen
MIPS (little endian only)	kvm, tcg
PPC	kvm, tcg
RISC-V	kvm, tcg
s390x	kvm, tcg
SPARC	tcg
x86	hvf (64 bit only), kvm, nvmm, tcg, whpx (64 bit only), xen

Tabla 4: Listado de Arquiteturas de CPU suportadas por QEMU [18]

Sin embargo, la utilización de emuladores o virtualizadores de memoria hace que no estén habilitadas todas las características de seguridad que puedan ofrecer los Entornos de Ejecución seguros y que existan problemas de dependencias con respecto a las librerías de los entornos a utilizar.

Por último, la última alternativa que se plantea en el presente proyecto es la **subcontratación de una máquina virtual** que se ejecuta dentro de un hardware compatible con los Entornos de Ejecución Fiables y con la extensión ya habilitada. Microsoft Azure [20] es uno de los portales que ofrece este tipo de servicios. Sin embargo, estas máquinas virtuales correr dentro de un hardware especifico, por lo que no entran dentro de la prueba de 12 meses gratis y hay que pagar el despliegue de las mismas. Estas máquinas son de pago por uso, es decir, que el coste mensual de la máquina se calcula en función del uso que se le haya dado, la memoria del hardware que se haya utilizado y de la RAM que necesita la máquina para ejecutar los programas internos.

6.1.3 Librerías TEE a utilizar

Una vez analizado los entornos y las diferentes alternativas posibles para la realización del despliegue de la solución, el siguiente paso es definir las librerías a utilizar.

La primera librería a describir es la librería **Linux-SGX** de Intel [21], que es una librería de código abierto en lenguaje C y C++ ofrecida por Intel. Esta librería ofrece el despliegue de aplicaciones dentro de enclaves seguros y, además, proporciona una biblioteca de criptografía de propósito general para aplicaciones Intel(R) SGX enclave. Sin embargo, este tipo de librerías sólo es compatible si se está ejecutando bajo un hardware compatible con la extensión SGX.

La segunda librería a describir se llama **openenclave** [22] el cual ofrece un SDK de código abierto destinado crear aplicaciones basadas en Entornos de Ejecución Fiables. Además, también da soporte a un conjunto de APIs que permiten construir e implementar aplicaciones en dentro de enclaves en múltiples plataformas tecnológicas, diferentes entornos desde la nube a híbridos a edge, y tanto para Linux como para Windows. Al igual que en el caso anterior, está escrito en lenguaje C y C++

La siguiente librería planteada es **Ego de edgelesssys** [23], la cual ofrece un framework con la finalidad de construir aplicaciones confidenciales en lenguaje Go. Estas aplicaciones confidenciales se ejecutan en enclaves cifrados y verificables en un hardware compatible con la extensión SGX de Intel. Ego simplifica el desarrollo de enclaves proporcionando dos herramientas sencillas de usar.

- **ego-go**, un compilador Go adaptado que construye ejecutables compatibles con enclaves a la vez que proporciona la misma CLI que el compilador Go original.
- **ego**, una herramienta CLI que gestiona todas las tareas relacionadas con enclave, como la firma y la creación de enclave.

Por último, la última librería es **QEMU-SGX de Intel** [24] que ofrece un emulador y virtualizador genérico y de código abierto de máquinas y espacio de usuario programada en lenguaje C. QEMU es capaz de emular una máquina completa en software sin necesidad de soporte de virtualización de hardware. QEMU también ofrece la posibilidad integrarse con los hipervisores Xen y KVM para proporcionar hardware emulado al tiempo que permite al hipervisor gestionar la CPU. Con el soporte del hipervisor, QEMU puede alcanzar un rendimiento casi nativo para las CPU. Sin embargo, es una librería a la que se ha dejado de dar soporte, por lo que es probable que la librería esté desactualizada y de problemas con las nuevas versiones de programas de Linux.

6.2 Criterios de evaluación

Para seleccionar la mejor o mejores alternativas que se han planteado en el punto anterior, se va a hacer uso de una serie de criterios de selección los cuales serán específicos para cada aspecto que se va a analizar. Para cada una de las opciones planteados, se definirán diferentes criterios de selección elegidos en función de las necesidades de cada criterio.

Además, con la intención de cuantificar las alternativas, se le asignará un coeficiente que refleja el grado de importancia a cada criterio y así obtener de forma numérica las mejores soluciones para desarrollar el presente trabajo y cumplir con sus objetivos. Los coeficientes no tendrán el mismo valor, ya que hay aspectos que tendrán una mayor importancia debido a las limitaciones del equipamiento utilizado para el trabajo.

6.2.1 Entornos de Ejecución Fiables con enclaves seguros

Para realizar dicha comparación y definir los criterios de selección, se hará de los siguientes artículos [10] [17] [25] que ya realiza una comparativa de las tecnologías SGX, AMD y ARM. Ya que la realización de los diferentes Entornos de ejecución Fiables no entra dentro del alcance del presente proyecto. Los artículos indicados definen los siguientes aspectos para realizar la comparación de las distintas tecnologías:

EL primer parámetro a tener en cuenta es la **seguridad ofrecida** por el entorno. En este criterio se tendrán en cuenta los diferentes mecanismos de seguridad que implementan los diferentes entornos a evaluar. Este parámetro es el más importante de todos, ya que el objetivo del presente trabajo es dotar al protocolo DNS de mecanismos de seguridad. Es por eso que este parámetro tendrá un **peso del 50%**.

Casos de Uso: Este parámetro determina el uso que tienen los Entornos de Ejecución Fiables y el tipo de aplicaciones que suelen ejecutarse dentro de estos entornos. Además, también se tendrá en cuenta los tipos de dispositivos en los que se pueden implementar los entornos. Dado que el objetivo principal del proyecto es la securización del protocolo DNS, a este criterio se le dará un valor **del 10%.**

Otro de los parámetros a tener en cuenta para la selección del entorno es el **rendimiento** que puedan ofrecer los entornos. El rendimiento es un aspecto importante de un TEE, ya que define el dominio de aplicaciones en el que se puede aprovechar un TEE concreto. Con este criterio, se plantea ver cuál es el entorno ofrece una mayor rapidez a la hora de la encriptación de datos para ver cual se pude llegar a utilizar en aplicaciones de tiempo real. A este criterio se le da un **coeficiente del 15%** debido a que, al tratarse de una consulta DNS, no es una aplicación que requiera una tasa alta de información a procesar y cifrar.

Otor de los parámetros importantes a tener en cuenta es el **uso de la CPU** a la hora de ejecutar una aplicación. Al estar incorporando mecanismos de seguridad al programa a a ejecutar, se hace un uso extra del uso de la CPU, ya sea para cifrar el mensaje o para autenticar a un programa externo. Este parámetro tiene un impacto notable, ya que al estar usando el protocolo DNS, es importante saber el número de recursos que se están utilizando por cada conexión DNS y así saber el un número máximo que pueden acceder al servicio y poder dimensionarlo. Es por eso que este apartado tendrá un **valor del 25%.**

6.2.2 Despliegue de la solución

Para la selección del despliegue de la solución, se van a tener en cuenta los siguientes parámetros.

El primer parámetro a tener en cuenta es la **Compatibilidad** que tenga el despliegue de la solución con las librerías EETs a utilizar en el proyecto. Es importante mirar las compatibilidades hardware y software que existen entre el despliegue y las librerías, ya que la incompatibilidad de dependencias puede hacer que la solución obtenida no sea la esperada. Por eso, se busca una ´despliegue que sea compatible con todas las opciones indicadas en el punto 6.1.3 Librerías TEE a utilizar. Por eso, a este criterio se le va a asociar un **coeficiente del 25 %**

Otro de los parámetros a tener en cuenta **Coste**. Este criterio tiene mucho valor porque hace que se dispare el presupuesto del proyecto. El aspecto económico es muy importante, ya que determina si un proyecto es viable o no. Es por eso que, en este proyecto, se va a buscar un equilibrio entre el coste del proyecto y la dedicación y el tiempo de instalación. A este criterio se le va a poner un **peso del 45%**.

Otro de los criterios a tener en cuenta es la facilidad y la flexibilidad de **Implementación** que ofrecen las diferentes soluciones a desplegar. Como se ha comentado previamente, una solución cuya implementación es muy compleja hace que se requieran de muchas horas de dedicación a

la puesta en marcha. Horas que acaba siendo trabajo de un ingeniero realizado y coste a añadir en el presupuesto. Es por eso que a este criterio se le añade un **coeficiente del 20%.**

Por último, está el criterio de **características ofrecidas**, que se centra en las características de seguridad ofrecen a la solución a realizar. Esto es debido a que una solución donde se virtualiza la memoria no se pueden implementar todas las características de los Entornos de Ejecución Fiables. Lo que puede limitar notablemente el grado de seguridad que se le ofrece al protocolo DNS. A este criterio se le añade un **valor del 10%**.

6.2.3 Librerías TEE

Para seleccionar la librería óptima a implementar en la solución final del presente trabajo, se seleccionan los siguientes criterios a evaluar:

El primer parámetro que se va a tener en cuenta es la **complejidad de instalación** que tienen las librerías, ya que esto puede generar la demora del tiempo planificado para la instalación de la librería y acortar el tiempo de realizar la solución. Además, si se utilizan librerías muy complejas, es posible que no se pueda optimizar los recursos de dichas librerías. Debido a esto, el valor que se le da a este **criterio es del 25%.**

Otro de los aspectos a tener en cuenta es la **flexibilidad** que puede ofrecer la librería a la hora de programar la solución. Que la librería permita realizar la solución con mayor flexibilidad permite que se pueda realizar el programa de diferentes formas y evaluar cual es la que ofrece un mayor grado de seguridad o realizar la solución implementando diferentes mecanismos de seguridad. Debido a esto, el valor que se le da a este **criterio es del 25%.**

Otro de los criterios importantes a analizar es la **compatibilidad** que tenga la librería con el entorno seleccionado y con las características del equipo a utilizar. Además, dado que se va a realizar una consulta DNS, también es importante utilizar librerías que permitan implementar un servidor DNS o un proxy DNS. ES por eso por lo que a este criterio se le dará un **valor del 30%.**

Por último, es necesario evaluar el **lenguaje de programación** con la que está diseñada la librería. El utilizar un lenguaje de programación no conocido puede llevar a gastar tiempo en aprender a programar en dicho lenguaje. Por eso, a este parámetro de le va a dar un **coeficiente del 20%.**

6.3 selección de alternativas

Una vez definidos los criterios de evaluación, el siguiente paso es da un valor en un rango de valores del 1 al 10 en función del rendimiento que muestren en los criterios, el cual se evaluará de forma descendente, es decir, el 10 lo mejor y 1 lo peor.

6.3.1 Entornos de Ejecución Fiables con enclaves

Como se puede observar en la siguiente tabla y, teniendo en cuenta los artículos donde se comparan las diferentes tecnologías [10] [17] [25], el Entorno de Ejecución Fiable que ofrece mayor seguridad son los enclaves de **Intel SGX.** Esto es debido al entorno aislado fiable que crea para almacenar la información y parte crítica del programa. Esto proporciona una pasarela de

enclave estrecha y protegida, aplica control de acceso a la memoria y aplica protección de integridad de la memoria, lo que lo convierte en un TEE adecuado para proteger cargas de trabajo que interactúan con datos sensibles desde el punto de vista de la seguridad [10]. Sin embargo, sí que utiliza más recursos que la Tecnología de Encriptado de Memoria y que la TrustZone de ARM y tiene mayores tiempos de procesado debido a que tiene que encriptar y desencriptar la información cuando accede a memoria. No obstante, dado que el protocolo que se va a securizar no requiere de altos flujos de información, es la tecnología óptima para cumplir con el objetivo del presente trabajo.

Criterio	Intel SGX	AMD MET	ARM TrustZone
seguridad ofrecida (50%)	**8**	6	5
Casos de uso (10%)	5	**7**	5
Rendimiento (15%)	7	**8**	8
Uso de la CPU (25%)	6	7	**8**
TOTAL	**7,05**	**6,65**	**6,2**

Tabla 5: Selección de alternativas de Entornos de Ejecución Fiables

6.3.2 Despliegue de la solución

A pesar de tener un coste mayor que la solución de virtualización completa (Memoria + enclave), la mejor solución que se ha determinado es la de alquilar una máquina virtual que se ejecute dentro de un hardware compatible con la tecnología a utilizar indicada en el subapartado anterior. Tras revisar el portal de Microsoft Azure, el alquiler de la máquina tiene un precio bastante razonable, siendo similar coste que tendría en horas de un ingeniero la realización del proceso de virtualización de memoria, procesador compatible con el TEE. Además, dado que es una máquina virtual destinada al uso de este tipo de tecnologías, por lo que la implementación de librerías y la compatibilidad con los enclaves es muy alta.

Criterio	Despliegue basado en Hardware	Despliegue basado en software	Alquiler VM
Compatibilidad (25%)	**9**	5	**9**
Coste (45%)	2	**10**	6
Implementación (20%)	8	**5**	9
Características ofrecidas (10%)	10	3	**8**
TOTAL	**5,75**	**7,05**	**7,55**

Tabla 6:Selección de alternativas de Despliegue de la solución

6.3.3 Librerías TEE

Por último, se ha decidido utilizar la librería que ofrece Ego para la implementación de aplicaciones confidenciales a través de enclaves. Esta librería es muy sencilla de implementar y de realizar la puesta en marcha. Además, a pesar de ser en un lenguaje poco conocido como el lenguaje Go, este permite de manera sencilla la implementación de distintas librerías compatibles a Ego, solo añadiendo un comando, que busca y añade la librería al fichero de configuración. Además, dada la forma de trabajar de la librería, se pueden incorporar cualquier programa dentro del enclave seguro y también ofrece mecanismos de atestación para que programas externos puedan interactuar con un enclave seguro.

Criterio	Linux-SGX	openenclave	Ego	QEMU-SGX de Intel
Instalación (25%)	6	5	**8**	3
Flexibilidad (25%)	6	**7**	**7**	6
Compatibilidad (30%)	**10**	**9**	**9**	7
Lenguaje programación (20%)	**6**	5	5	**6**
TOTAL	**7,2**	**6,7**	**7,45**	**5,55**

Tabla 7: Selección de alternativas de Librerías TEE

6.4 Resumen y conclusiones

Para concluir con este apartado, se hacer un breve resumen de cómo quedará el desarrollo de la solución para cumplir con el objetivo de securizar el protocolo DNS: se hace uso de Entornos de Ejecución Fiables de Intel SGX a través de los fichero y recursos que ofrece la librería EGo programada en lenguaje Go. Además, el despliegue se realizará en una máquina virtual contratada utilizando el portal Microsofl Azure.

7 ANÁLISIS DE RIESGOS

En el presente apartado se analizan los posibles inconvenientes que puedan surgir a la hora de realizar el desarrollo del proyecto e influir de forma negativa al mismo. Esto puede provocar la demora del desarrollo y, como consecuencia, no cumplir con las fechas e hitos planteados en la planificación. Debido a esto, es necesario realizar un estudio para detectar los posibles contratiempos que puedan aparecer a la hora de la realización del presente proyecto y planificar un plan de contingencia para solucionar dichos problemas en un tiempo aceptable y no comprometer las fechas de entregas planificadas.

Para la realización del análisis, en primer lugar, se hace una identificación de riesgos. Luego, mediante el uso de la matriz probabilidad-Impacto, se realiza un análisis para ver cuál es el riesgo que es más frecuente de que ocurra y ver cuál es el que tiene un mayor impacto en el desarrollo de la solución. Por último, como ya se ha comentado, se realiza un plan de contingencia que permita tomar medidas preventivas para reducir el impacto que puedan tener en la solución.

7.1 Identificación de Riesgos

7.1.1 Selección de la información errónea

Uno de los primeros riesgos que se ha detectado en el presente proyecto es la **selección de la información** concreta necesaria. Los enclaves SGX es un mecanismo relativamente reciente, por lo que es posible que no haya mucha información acerca del uso del mismo o de las aplicaciones que puede ejecutar dentro del mismo.

Sin embargo, el verdadero riesgo existente en este proyecto reside en la complejidad que tenga la información buscada acerca de los enclaves. Estos Entornos de Ejecución Fiables tienen un alto grado de seguridad y ofrecen estrictos mecanismos de seguridad que, para poder manejar bien un enclave, hay que conocer bien el funcionamiento y despliegue de dichos mecanismos. Por eso, la búsqueda de información sin el conocimiento de estos mecanismos puede ser un proceso más costoso de lo normal, proporcionando demoras en las fechas de la planificación.

7.1.2 Utilización de Hardware que no dispone de SGX

Uno de los mayores riesgos que tiene el presente trabajo es el despliegue de la solución mediante un **equipamiento que no dispone de la extensión SGX** de Intel para poder almacenar un programa o parte de su código más crítico en un enclave. Este riesgo es bastante problemático, ya que muchas de las librerías utilizadas para el uso de SGX enclaves tienen como requisito que el equipo en el que se vaya a ejecutar el programa disponga de la extensión SGX activa.

Este riesgo es uno de los más críticos, ya que utilizar un equipo sin la extensión ni recursos necesarios, imposibilitaría el desarrollo de la solución por completo y paralizaría el presente trabajo.

7.1.3 Error de Acceso a la máquina virtual

Otro de los riesgos notorios que tiene el presente proyecto es tener **problemas de acceso a la máquina virtual** que se esté desplegando la solución. Como se ha comentado en el punto anterior, la solución se realizará utilizando una máquina virtual de Linux. En algún momento, es posible que esa máquina no esté disponible o haya problemas con el acceso a la misma.

Este es otro riesgo a tener en cuenta, ya que al igual que ocurría con el riesgo anterior, sin acceso a la máquina virtual se paralizaría por completo el desarrollo de la solución haciendo imposible cumplir con las fechas indicadas en la planificación.

7.1.4 Incompatibilidad de librerías

El uso de una máquina virtual de Linux y el uso de librerías o repositorios para desplegar la solución puede generar incompatibilidades a la hora de ejecutar alguna de dichas librerías. Esto es debido a que estas pueden estar obsoletas respecto a la versión de Linux que se esté utilizando y haga que algunas de las funcionalidades de las librerías no puedan ser utilizadas. También puede ocurrir el caso contrario, es decir, que se utilice una versión inferior de Linux y que las librerías se hayan actualizado, siendo incapaz de utilizar las librerías en esa versión obsoleta de Linux. A este riesgo se le denomina la incompatibilidad de librerías y es un riego a tener muy presente en este trabajo.

Otro de los problemas es que, juntado una versión obsoleta de Linux para utilizarlo con una librería concreta haga que la máquina virtual carezca del soporte de seguridad y corrección de errores necesario, por lo que sea incompatible el uso de dicha librería.

7.2 Análisis de los riesgos identificados

Una vez identificados los riesgos, el siguiente paso es realizar un análisis para determinar el grado de impacto que tienen en el desarrollo de la solución en caso de que se presente uno de estos riesgos. Además, también se tendrá en cuenta la probabilidad con la que se pueda dar dicho riesgo.

El análisis consta de dar un valor a la probabilidad que tenga dicho riesgo de darse en la realización del presente proyecto y otro valor para medir el nivel de impacto que tendría en la solución propuesta. Respecto a los riegos descritos previamente, la siguiente tabla muestra la los valores que se le han asociado a cada uno de ellos:

Riesgo	Probabilidad	Impacto
Selección de la información errónea (R1)	Medio	Bajo
Hardware sin extensión SGX (R2)	Medio	Alto
Error de acceso a la VM (R3)	Alto	Alto
Incompatibilidad de librerías (R4)	Medio	Medio

Tabla 8: Características de los riesgos

Para realizar la evaluación de los riesgos, se utilizará la Tabla 9 definida como matriz probabilidad-impacto de 3x3 en la que tanto la probabilidad como el impacto se van a medir en función de tres valores diferentes: **bajo** el cual tendrá un coeficiente del 0.2, **medio** el cual

tendrá un coeficiente del 0.5 y **alto** el cual tendrá un coeficiente del 0.8. La siguiente tabla muestra el valor de la multiplicación de la matriz probabilidad impacto de cada uno de los riesgos:

Probabilidad	Bajo	0,04	0,1	0,16
	Medio	**R1** 0,1	**R4** 0,25	0,4
	Alto	0,16	**R2** 0,4	**R3** 0,64
		Bajo	Medio	Alto
	Impacto			

Tabla 9: Matriz probabilidad-impacto

7.3 Plan de contingencia

Una vez definida la el grado de importancia que tienen cada uno de los riesgos, el siguiente paso a realizar es la creación de un plan de contingencia para buscar la solución óptima en caso de se produzca uno de estos riesgos.

Empezando con el primer riego, la **selección de la información errónea** es un riesgo que puede ocurrir varias veces a la hora de la realización de la búsqueda de información. No obstante, es un riesgo que apenas tiene impacto en el proyecto, ya que solo retrasaría la tarea de búsqueda y selección de información durante unas horas. Es por eso que se toma la decisión de **asumir el riesgo** y no plantear ninguna medida si ocurre este problema a lo largo de la realización del trabajo.

El riesgo de **utilización de un hardware que no disponga de la extensión SGX de Intel**, es uno riesgo bastante notable que puede afectar de forma significativa al desarrollo de la solución. El problema de este riesgo es que no disponer de un hardware compatible con SGX limitaría considerablemente las librerías de enclaves a utilizar y haciendo necesario una búsqueda exhaustiva de librerías que permitan su uso en el hardware a utilizar o de buscar herramientas que permitan virtualizar una memoria para su uso con enclaves. Este último proceso es bastante complejo, ya que es necesario adquirir el conocimiento del funcionamiento interno de las memorias, además que el poder desplegar la memoria acaba consumiendo una gran cantidad de tiempo, haciendo notorios los atrasos respecto al presupuesto. Es por eso por lo que la solución planteada para este riesgo es la **contratación de una máquina virtual en un servicio en la nube que corra sobre un hardware que tenga habilitada la extensión SGX de Intel**.

En consecuencia, del riesgo anterior, se ha detectado que el error más crítico que pueda ocurrir en el presente trabajo es no un **error de acceso a la máquina virtual** debido a un problema de indisponibilidad o porque el mecanismo de acceso utilizado no se encuentre disponible. Este problema es verdaderamente crítico ya que, si ocurre algún caso de estos, sería inviable la continuación del proyecto. Es por eso que la **solución preventiva que se ha tomado para este riesgo es contratar una máquina virtual a través del portal de Microsoft Azure** y desplegar una máquina virtual en la nube de Azure. El alquiler de este tipo de servicios no es un coste muy elevado a pagar mensualmente y además ofrece diferentes niveles de seguridad y disponibilidad para poder acceder en cualquier momento a la máquina virtual. Además, por defecto la máquina virtual de Azure solo tiene habilitado la conexión a través del protocolo SSH. Por lo que otro

mecanismo preventivo que se le ha asociado a este riesgo es la habilitación de un segundo método de acceso, que en este caso se realizará a través del protocolo RDP (Remote Desktop Protocol). De forma que se pueda acceder a la máquina virtual a través de dos métodos diferentes.

Por último, el último riesgo tiene un nivel moderado pero que hay que tenerlo en cuenta ya que dificultaría bastante el despliegue de la solución. Este riesgo es la **incompatibilidad de las librerías** a utilizar en la máquina virtual. Es por eso que se tomará una medida preventiva y es la **búsqueda de información en documentación y páginas de Microsoft Azure de las diferentes librerías compatibles con la máquina virtual** contratada.

8 ESTUDIO DE SEGURIDAD

En este apartado se realiza un estudio de seguridad previo con la finalidad de diseñar la solución más segura para la consulta DNS a realizar. El estudio de seguridad se hará en base a la normativa ISO 31000:2018 Gestión del riesgo – Directrices [26]. La siguiente imagen muestra el proceso para realizar una gestión de riesgos:

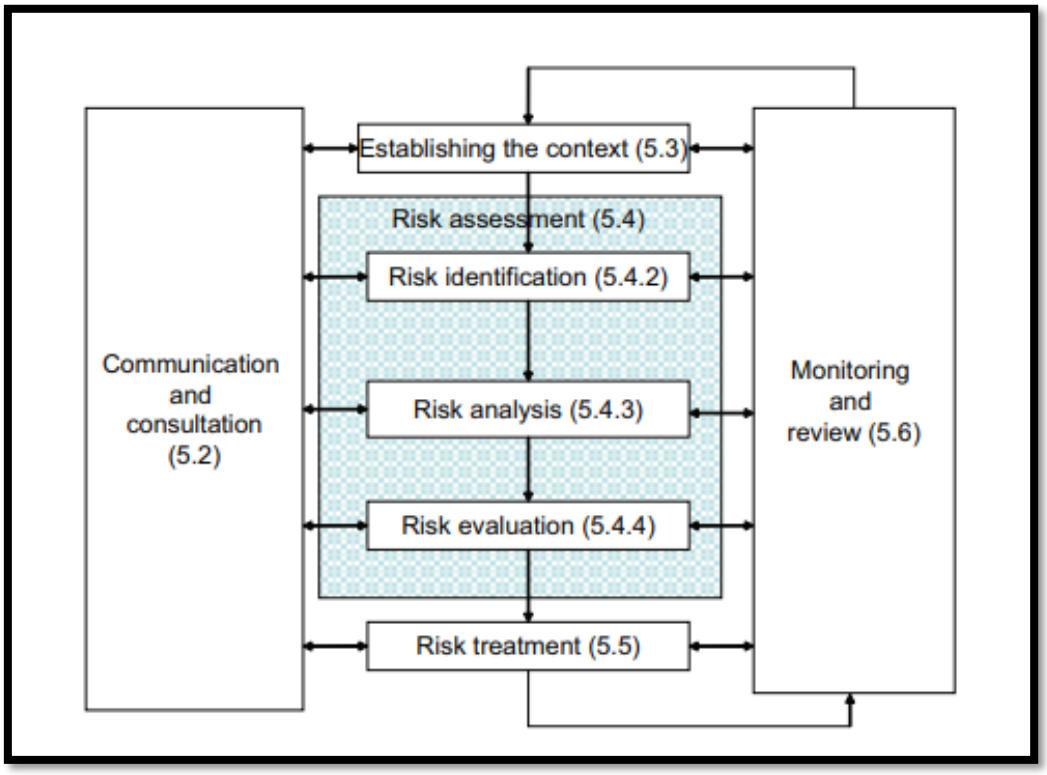


Figura 16: Proceso de Gestión de Riesgos [26]

Para realizar el estudio de seguridad, primero se efectúa un análisis de riesgos que puede tener la solución final a realizar y ver el impacto y las consecuencias que puede tener la explotación de dichas vulnerabilidades. Luego se realizará una evaluación de riesgos para ver qué riesgos son más críticos y más pueden afectar a esa solución final. Por último, se realizará un plan de mitigación, donde se indican medidas que se van a tomar para mejorar la seguridad de los activos del proyecto y mitigar las consecuencias a la hora de que el desarrollo sufra un ataque de seguridad.

8.1 Análisis de riesgos

Como se ha comentado en la introducción de este apartado, el primer paso es la realización de un análisis de riegos con la intención de definir los activos del proyecto, los riesgos o ataques que pueden sufrir y el impacto que tendría dicho ataque en la solución a desarrollar.

8.1.1 Identificación de activos

En primer lugar, el activo principal que existe en el presente trabajo es la **información que forman la consulta DNS**, tanto el nombre de dominio como la IP asociada al mismo. Otro de los activos importantes a proteger es el bloque del **proxy DNS**, ya que se encarga de obtener la petición DNS del cliente y realizar la consulta a un servidor externo. Por otro lado, otro de los activos a proteger son las **comunicaciones** creadas para conectar los diferentes programas que formar la solución desplegada. Las comunicaciones existentes son las siguientes:

- La conexión socket para la transmisión de información entre el programa cliente y el proxy DNS.
- La comunicación entre el programa ejecutado fuera del enclave y el servidor externo DNS que se encarga de realizar la respuesta DNS.

8.1.2 Identificación de riesgos

Una vez definidos los activos existentes, el siguiente paso a realizar es la identificación de los riesgos que pueden tener. El riesgo principal que puede tener una consulta DNS es la **manipulación de la consulta DN**S y que al usuario le llegue una respuesta que no sea la dirección IP verídica del nombre host que ha solicitado. Esto afectaría a la información que se maneja en la respuesta DNS.

Existen diferentes posibilidades para que ocurra dicha manipulación. La primera de ellas es la **usurpación de identidad** y que el programa cliente, el que maneja el usuario, se conecte a un servidor DNS o proxy DNS fraudulento que devuelva una dirección IP falsa del nombre de dominio para que el usuario se conecte a otra página web. Este tipo de ataque afectaría tanto al activo de la información, como al activo de la comunicación cliente-servidor que se realiza, porque se está tanto manipulando la información como alterando la comunicación, realizando el cambio del destinatario de la petición DNS realizada por el cliente.

Otro ataque puede ser un riesgo bastante notable en la solución es el ataque de Hombre en el Medio o **"Man in the Middle"**. Este ataque se basa en que una tercera entidad pueda interceptar el mensaje DNS y cambie el contenido de este y se vea afectada la integridad del mensaje. En este caso, también se comprometerían los mismos activos que con el ataque anterior.

Estos dos tipos de ataques serían los ataques más críticos a los que se puede enfrentar la solución planteada en el presente proyecto. No obstante, también existen otros tipos de ataques que, a pesar de que no afecten al contenido del mensaje de respuesta DNS, también hay que tenerlo en cuenta ya que afecta al rendimiento de la solución y compromete las comunicaciones DNS. Este tipo de ataques se denominan **ataques de denegación de servicios** y, debido a que el protocolo dispone de un tiempo de espera definido, hace que el cliente no reciba respuesta a la petición que ha llegado a realizar. En este caso, el activo que se vería afectado sería el proxy DNS, ya que el objetivo del ataque sería inhabilitar el servidor para que no pueda manejar las peticiones que se le realiza.

8.1.3 Impacto de los riesgos

Para medir el impacto que pueden tener los riesgos en los activos, primero hay que analizar la posibilidad que existe de que una consulta DNS sea víctima de uno de los ataques indicados anteriormente.

Respecto a los ataques, el que más probabilidades tiene de que ocurra es el ataque de "Man in the Middle". En este escenario, como no es necesario realizar una manipulación de la conexión entre el cliente y el servidor, sino que se monitorizan los mensajes y la información que circulan en la conexión realizada entre el cliente y servidor, pudiendo hasta manipular la información del mensaje. Además, este tipo de ataques, como y se ha comentado, pueden cambiar el mensaje y enviar al cliente una dirección IP fraudulenta. El impacto que tendría en la consulta sería muy importante, ya que se estaría accediendo a una dirección IP que no está relacionada con el nombre de dominio indicado en el mensaje de petición.

El ataque de usurpación de identidad es menos probable que ocurra que el de "Man in the Middle". Esto es debido a que, para poder realizar el ataque, hay que cambiar la configuración de conectividad del equipo del usuario para que realice una conexión con el servidor fraudulento. De esta manera, el usuario realizaría la petición DNS con el servidor maligno y recibiría una dirección IP fraudulenta que no se asemeja con la IP pedida. Este tendría el mismo impacto que el ataque anterior.

Por último, respecto al ataque de denegación de servicios, es un ataque que no llega a comprometer la información del mensaje de respuesta DNS, pero es un ataque que impide al cliente la respuesta a la petición realizada. Es por eso por lo que es un riego más moderado que los otros dos. No obstante, también hay que tenerlo en cuenta, ya que es un ataque que afecta a la disponibilidad del servidor.

8.2 Evaluación de riesgos

En este subapartado se realiza una evaluación de los riesgos que pueden comprometer seguridad de la solución de la solución en función del impacto que tienen cada uno de los riegos analizados en el subapartado anterior.

El impacto que puede generar el ataque "Man in the Middle" es bastante notorio y comprometería mucho la seguridad del sistema. Además, como se ha comentado previamente, es un ataque bastante factible de realizar ya que, con un capturador de paquetes, se pueden ver todos los paquetes que viajan a través de un enlace. Esto hace que haya que tomar medidas para proteger las comunicaciones entre usuario y el proxy y el proxy y el servidor externo.

Respecto al ataque de usurpación de identidad, el impacto que tiene sufrir uno de estos ataques también es crítico. Al igual que en el caso anterior, a pesar de ser menos probable, también es necesario proteger las comunicaciones entre el cliente y los servidores. Además, también es necesario proteger el proxy DNS mediante mecanismos de autorización y control de acceso.

Por último, el impacto que tiene el ataque de denegación de servicios es un impacto más moderado que los otros dos. Sin embargo, hay que tenerlo en cuenta, ya que se puede dar el caso de que el servidor deje de estar disponible durante periodos de tiempo notables. En este caso, se ha planteado dotar al servidor proxy con protocolos de control de acceso para conseguir que las peticiones al proxy se limiten e impidan saturar el servidor y dejarlo inhabilitado.

8.3 Plan de mitigación de riesgos

En este apartado se realiza un plan de mitigación de riesgos, de forma que se consiga reducir tanto la probabilidad de sufrir el ataque como del impacto que pueda ocasionar el mismo.

En este caso, se van a plantear soluciones para los ataques de usurpación de identidad y ataques de “Man in the Middle”, se ha plateado dotar a las conexiones entre el proxy DNS y el servidor externo de mecanismos de cifrado. De esta forma, se consigue dotar a las conexiones de integridad y confidencialidad. Esto se realizará haciendo uso del protocolo TLS (Transport Layer Security) o Seguridad de la Capa de Transporte y siempre se realizarán las consultas al servidor DNS de Google 8.8.8.8. Así, el servidor externo tendrá que compartir su certificado firmado por una Autoridad de Certificación (CA) y evitarlas vulnerabilidades del protocolo DNS ante ataque de usurpación de identidad y “Man in the Middle”.

Respecto a la comunicación entre el usuario y el proxy, con la intención de ofrecer mecanismos de integridad y confidencialidad a los dos extremos, también se realizará mediante una comunicación TLS. Sin embargo, se hará uso de los **enclaves seguros** SGX de Intel para ofrecer un mayor grado de seguridad. Como se ha comentado previamente los enclaves ofrecen un entorno de ejecución aislada de forma que ningún usuario no autorizado pueda acceder ni enviar ningún tipo de información dentro de él. El proceso de acceso al enclave se denomina proceso de atestación y requiere de la identificación de los programas o entidades que quieren acceder al enclave. Además, antes de realizar el proceso de atestación, lo primero que se necesita es una clave de acceso al enclave. Lo que hace que genere una capa extra de seguridad, haciendo más complejo el acceso al servidor proxy y la información que este está manejando dentro del enclave.

De esta forma se consigue securizar el protocolo DNS protegiendo la consulta de las principales vulnerabilidades que se le han detectado.

9 DESCRIPCIÓN DE LA SOLUCIÓN

En este apartado se presenta y analiza la solución principal que se ha tomado para satisfacer los objetivos planteados del presente documento. El proyecto se basa en la creación de un entorno de programación aislado para hacer que un cliente pueda hacer una consulta DNS de forma segura. En el siguiente diagrama, se puede observar el sistema a desarrollar para satisfacer la petición DNS del cliente:

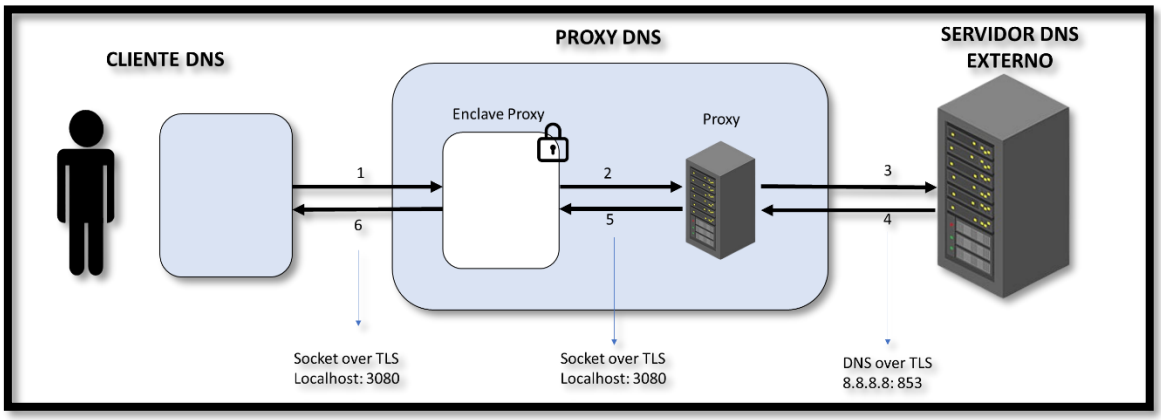


Figura 17: Esquema general de la solución propuesta

9.1 Solución propuesta

Como se ha comentado previamente, la solución tiene como objetivo la securización del protocolo DNS mediante el uso de un entorno de ejecución fiable. Para ello, se hace uso de los enclaves seguros de Intel SGX, los cuales crean un entorno de ejecución aislado que ofrece una mayor seguridad de sobre la información que se maneja dentro de los enclaves.

Tal y como se indica en el apartado anterior, debido a las características especiales que se necesitan en los equipos para desplegar una aplicación haciendo uso de enclaves seguros, se hace uso de una máquina virtual de Microsoft Azure. Esta máquina virtual se ejecuta en un hardware compatible con enclaves. La siguiente imagen muestra las características que tiene la máquina virtual utilizada para desarrollar la solución.

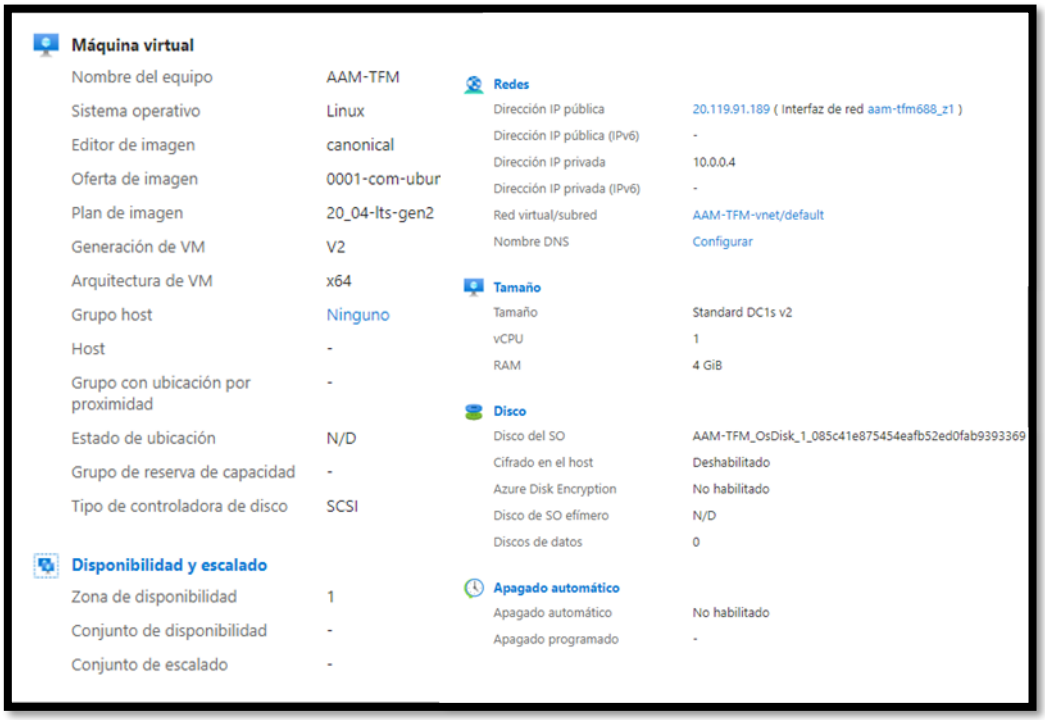


Figura 18: Características máquina virtual de Microsoft Azure

El acceso a la máquina virtual se puede realizar de das formas diferentes: por protocolo SSH o por protocolo RDP (Real Desktop Protocol) o Protocolo de Escritorio Remoto. Para el caso de este trabajo se ha accedido a la máquina virtual a través del protocolo RDP haciendo uso del puerto 3389. Para ello, la siguiente página web muestra un tutorial el cual sirve para activar la posibilidad de conectarse a la máquina virtual a través del protocolo RDP [27].

Una vez instalada la máquina virtual y con el acceso configurado, el siguiente paso es la implementación y descarga de librerías para poder desarrollar aplicaciones haciendo uso de enclaves seguros. En este caso, se ha hecho uso de la biblioteca EGo de Edgeless systems [21-23] la cual ofrece un framework para crear aplicaciones confidenciales en lenguaje go y la forma de compilar dichas aplicaciones. El repositorio ofrece las dependencias necesarias para realizar aplicaciones dentro de enclaves dentro de un hardware compatible con Intel SGX. La siguiente página contiene la información ubicada en el fichero “README.md”, en el que se detallan los pasos a seguir para la correcta configuración de las dependencias en la máquina virtual [28].

Una vez configurado el entorno, el siguiente paso es la realización de la aplicación para cumplir con los objetivos propuestos en el presente proyecto. Dado que el programa que se ejecuta dentro de un enclave está aislado y no permite la salida de información al exterior, se ha dividido la aplicación en tres bloques:

- El cliente que realiza la petición DNS.
- El servidor Proxy que se encarga de gestionar la consulta DNS con el exterior.
- El servidor DNS externo que devuelve la dirección IP del nombre de dominio pedido.

9.1.1 Cliente

El primer bloque, es un programa que se encarga de realizar la petición DNS al proxy. Para ello, el primer paso es crear la conexión entre el cliente y el proxy DNS que, en este caso, dicha conexión se realiza con la parte del proxy ubicada dentro del enclave, tal y como se muestra en la Figura 17.

La razón detrás de este planteamiento tiene la finalidad de evitar un ataque de usurpación de identidad, es decir, que el cliente se conecte a un proxy que se haga pasar por el servidor al que el cliente desea conectarse y que este le responda con una dirección IP fraudulenta que no coincide con el nombre de dominio pedido.

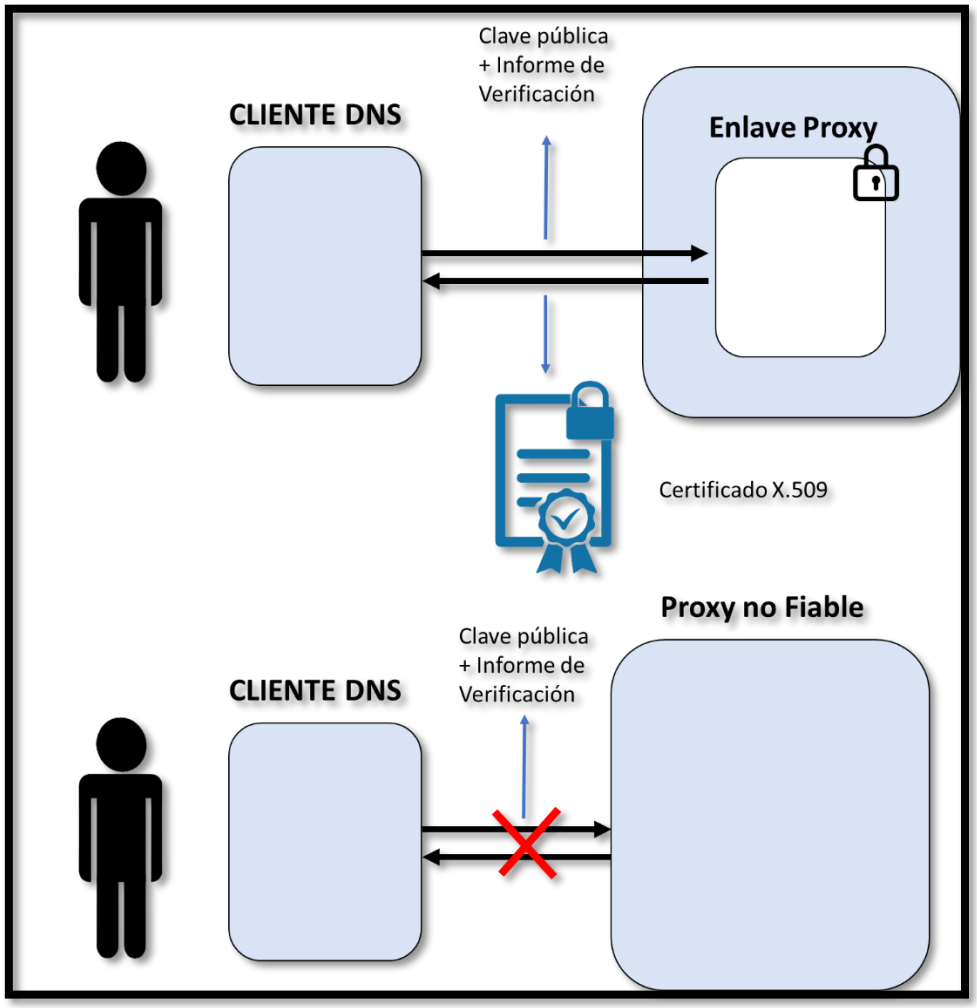


Figura 19: Esquema general de la solución propuesta

Esto es posible debido a que, como se ha comentado previamente, un cliente tiene que pasar por un proceso de atestación para poder comunicarse con el enclave. Además, previo al proceso de atestación, el cliente debe disponer de la clave pública del enclave para poder acceder al enclave. Esto proporciona un mayor grado de seguridad al desarrollo. Por lo que este método hace que sea más seguro el envío de la petición DNS a través del enclave que directamente al proxy DNS.

Una vez atestado el cliente de forma exitosa, el enclave y el cliente crear una variable de configuración TLS con la información del proceso de atestación y crea una comunicación socket cifrada a través del protocolo TLS. Este canal seguro permite transmitir y recibir información confidencial entre el cliente y el enclave que, en este caso, será el nombre de dominio a traducir por el servidor proxy.

La otra funcionalidad que tiene el cliente es la recepción de la dirección IP a través del canal seguro creado por el enclave y mostrar dicha información por pantalla.

9.1.2 Proxy DNS

La aplicación a desarrollar tiene que ser capaz de realizar las consultas y respuestas DNS. Para ello, ha sido necesario el despliegue de un proxy DNS que sea capaz de gestionar la petición del cliente e interacción con servidores externos. Debido a que la aplicación esta realizada en lenguaje go, se ha hecho uso del repositorio de github miekg/dns [29], el cual ofrece una API con librerías DNS en go. Gracias a estas librerías se ha desplegado el proxy DNS que interactúa con un servidor público DNS para obtener la dirección IP del nombre de dominio pedido por el cliente.

La librería EGo ofrece una forma sencilla de creación de aplicaciones dentro de un enclave. No obstante, la forma que tiene la librería de funcionar es ejecutando una aplicación entera dentro del enclave. Además, al intentar ejecutar el proxy dentro del enclave, se presentaba imposible el interactuar con un servidor externo, ya que este no disponía de la dirección pública del enclave ni de los certificados a compartir.

A consecuencia de esto, se ha tenido que dividir la aplicación proxy en dos programas, de forma que el programa ubicado dentro del enclave se encarga la interacción con el cliente DNS y el otro programa se encargaba de la interacción con el servidor DNS externo.

9.1.2.1 Enclave

Debido a las características y los mecanismos de seguridad que ofrecen los enclaves, dentro del enclave se añadirá el código del proxy más sensible y la información a transmitir al cliente DNS. En este caso, al igual que la interacción con el cliente, también se creará una comunicación socket sobre TLS para recibir la información que recibe el proxy DNS del servidor externo. De este modo, la comunicación entre los dos programas que completan el servidor proxy será realizará de forma segura.

Uno de los problemas que tienen los enclaves, tal y como se vio en el estado del arte, es la cantidad de recursos de CPU que necesita para funcionar. Por este motivo, dentro del enclave sólo se ejecutará la transmisión de la información tanto de la petición como de la respuesta DNS. De esta manera, en el enclave se recibirá el nombre host del cliente fiable que desea realizar la petición y se lo transmitirá al proxy DNS para que realice dicha consulta DNS con un servidor externo. Y, cuando el proxy reciba la respuesta de la consulta, este se lo pasará al enclave a través del canal seguro y lo reenviará al cliente a través de la conexión TLS realizada.

Una vez compilado el código del enclave, el siguiente paso a realizar es la firma del enclave. Este paso es realmente importante, ya que realiza la creación del enclave y la creación de un par de claves pública y privada utilizadas para el acceso al enclave.

Figura 20: Firma del enclave y creación de claves para el acceso

En conclusión, de la forma en la que está planteado el programa en el interior del enclave, se están consiguiendo los siguientes mecanismos de seguridad:

- El primer mecanismo que ofrece es la autenticación y autorización del programa externo. Esto se realiza mediante un proceso de atestación en el que el cliente envía su certificado creado mediante el estándar X.509. De forma que sólo clientes o programas legítimos pueden acceder a la información que se está ejecutando en el enclave.
- Ofrece integridad y confidencialidad de forma que en la comunicación entre el servidor y el programa legítimo se realice de forma segura.

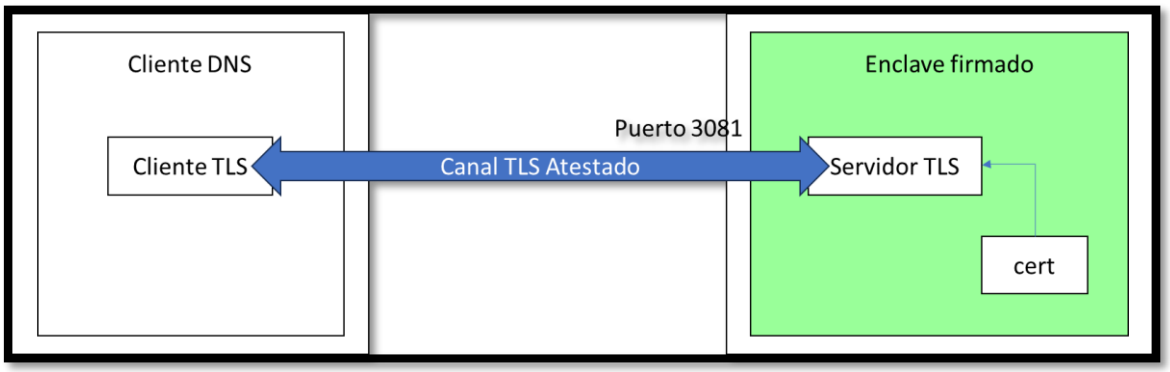


Figura 21: Esquema de proceso de conexión entre cliente externo y enclave

9.1.2.2 Proxy DNS

La segunda parte del proxy DNS se basa en la recepción de la petición del cliente a través del enclave seguro y la realización de la petición del cliente.

En esta parte de la solución, el programa interactúa con el enclave para recibir el nombre de dominio para hacer la petición DNS al servidor externo y, en consecuencia, recibir la respuesta de la petición y reenviar la respuesta al cliente a través de, enclave.

Para ello, se conecta al enclave a través de un proceso de atestación y recibe el nombre de dominio a través de la comunicación socket segura iniciada por el enclave.

Una vez que recibe la información del enclave, el programa genera una comunicación DNS a través del protocolo TLS con el servidor DNS de Google 8.8.8.8. La comunicación se realiza a través del puerto 853, el cual es un puerto “well known” que se encarga de transportar tráfico DNS encapsulado en mensajes TLS.

La comunicación entre el proxy y el servidor DNS de Google se inicializa cuando el servidor externo envía su certificado al proxy para dotar la comunicación del mecanismo de integridad. Una vez verificado el certificado del servidor, se negocian las claves para realizar el cifrado de información y añadir confidencialidad a la comunicación. Una vez creada la comunicación y cada extremo teniendo la clave correspondiente, se cifran los mensajes de la comunicación para proteger al protocolo de la vulnerabilidad a ataques de “Man in the Middle” que tiene.

Una vez recibida la respuesta a la petición DNS, como se ha comentado previamente, el proxy DNS envía la información a través del socket al enclave para devolver la dirección IP al cliente.

9.2 Diseño de alto nivel

Para facilitar el entendimiento de la solución desarrollada, se realiza un diseño de alto nivel de la solución desarrollada. Los diseños de alto nivel están basados en diagramas UML (Unified Modelling Languaje). Más concretamente se realizarán los diagramas de casos de uso, diagramas de estados y diagramas de flujo de la solución.

9.2.1 Diagrama de casos de uso

La finalidad del diagrama de casos de uso es la representación de la interacción de un usuario con el desarrollo de la solución planteada que, es este caso, lo realiza mediante el programa del cliente. Además, se tienen en cuenta la interacción existente que hay entre los actores que participan en la securización del protocolo DNS.

Como se puede observar en la Figura 22, se han identificado cuatro actores. Tres de ellos se basan en los programas que se han realizado para la consulta DNS y lego está el servidor externo DNS que es el que se encarga de dar respuesta a la petición realizada por el servidor proxy.

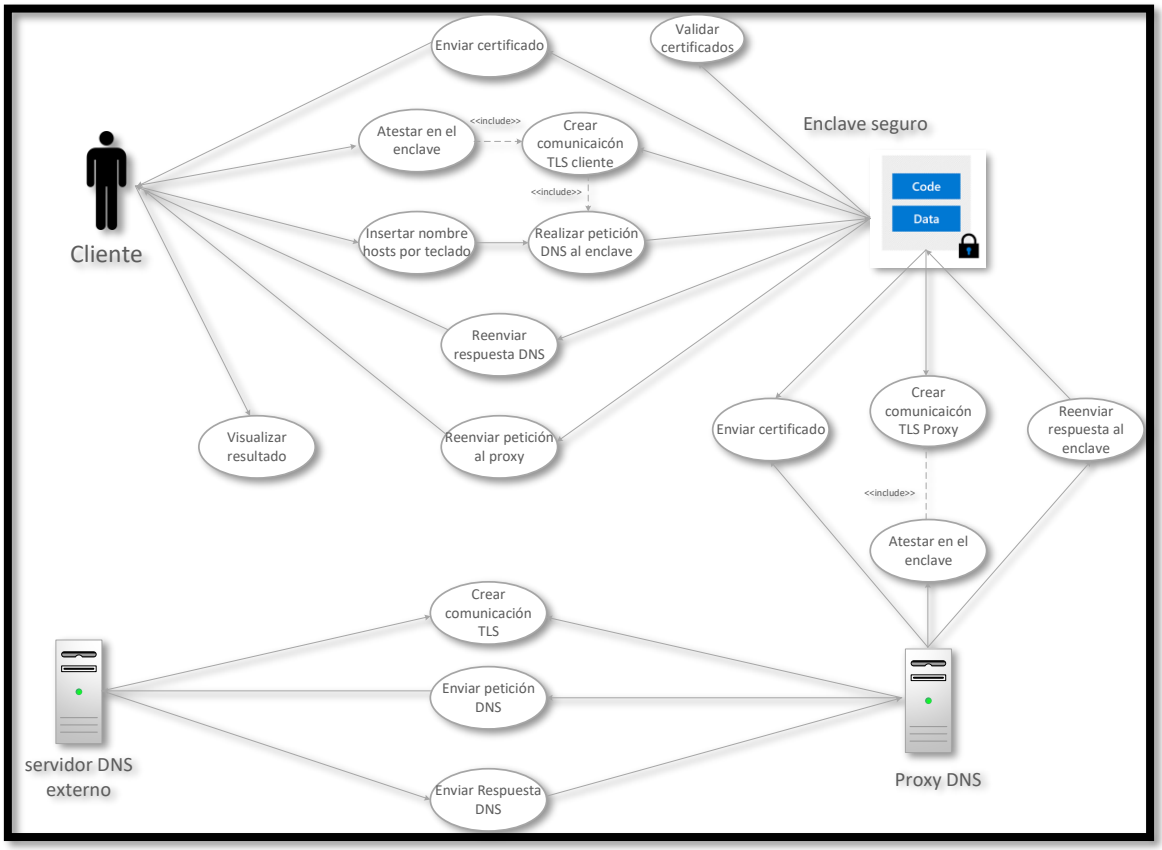


Figura 22: Diagrama de Casos de Uso

Por un lado, se observa la interacción del usuario cliente y el proxy con el programa interno del enclave para poder seguir con la aplicación. Esta interacción se realiza para asegurar que los programas externos que quieren acceder a la información que maneja el enclave son fiables.

Tras este proceso, lo siguiente que se aprecia en el diagrama es que la función del programa ejecutado en el enclave es el reenvío de información de un extremo a otro. Por último, cabe destacar que el programa del proxy es quien interactúa con el DNS externo de Google.

9.2.2 Diagrama de estados

El siguiente diagrama que se ha desarrollado es el diagrama de estados, el cual se basa en la representación de los distintos estados que puede tener el programa. Dado que la solución está basada en tres programas diferentes, se ha realizado un diagrama de estados por cada programa. A continuación, se muestran los tres diagramas de estados que se han planteado.

9.2.2.1 Programa cliente

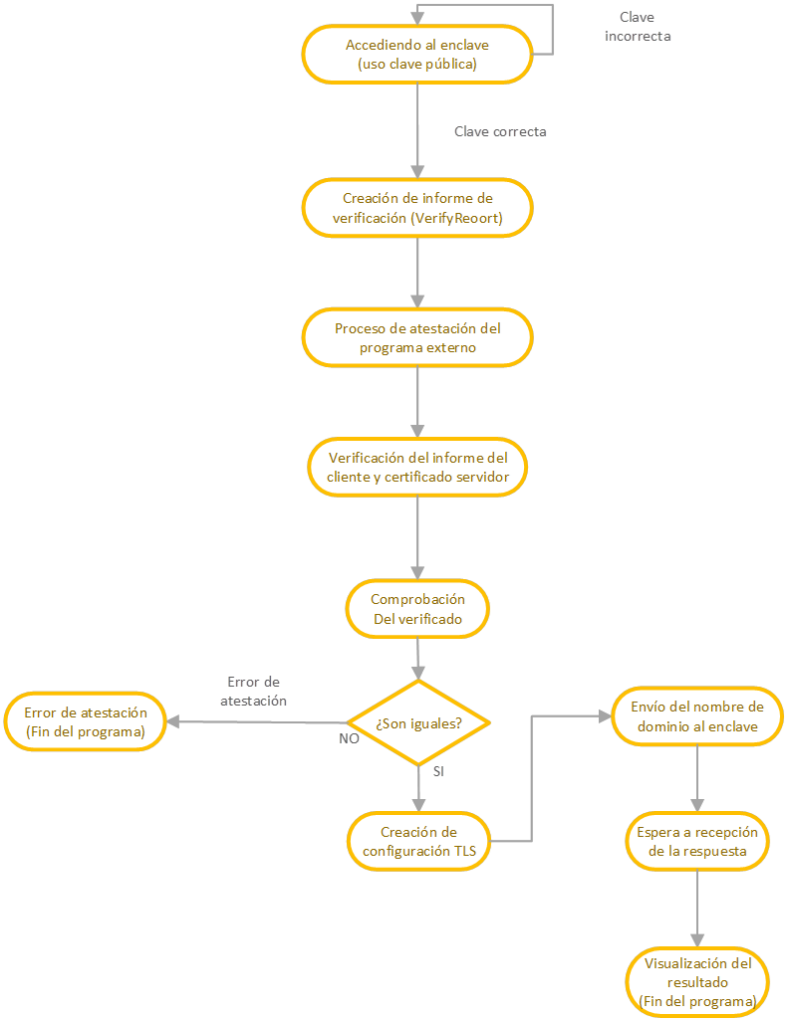


Figura 23: Diagrama de Estados del programa cliente

El diagrama muestra las diferentes fases por las que pasa el programa. La primera fase se basa en el proceso de atestación que tiene que realizar el cliente para poder acceder al programa ejecutado en el enclave., en este se comprueba el informe de verificación realizado por el cliente con el del servidor para la creación de la comunicación TLS entre ambos.

Luego, se pasa a los estados relacionados con el envío y recepción de mensajes a través de la comunicación TLS creada que, en este caso, se basa en el envío del nombre de dominio a resolver y la recepción IP asociado a dicho nombre El último estado en el que se encuentra el programa es la visualización de la respuesta DNS al nombre y dominio que ha solicitado.

9.2.2.2 Programa Proxy ejecutado en el enclave

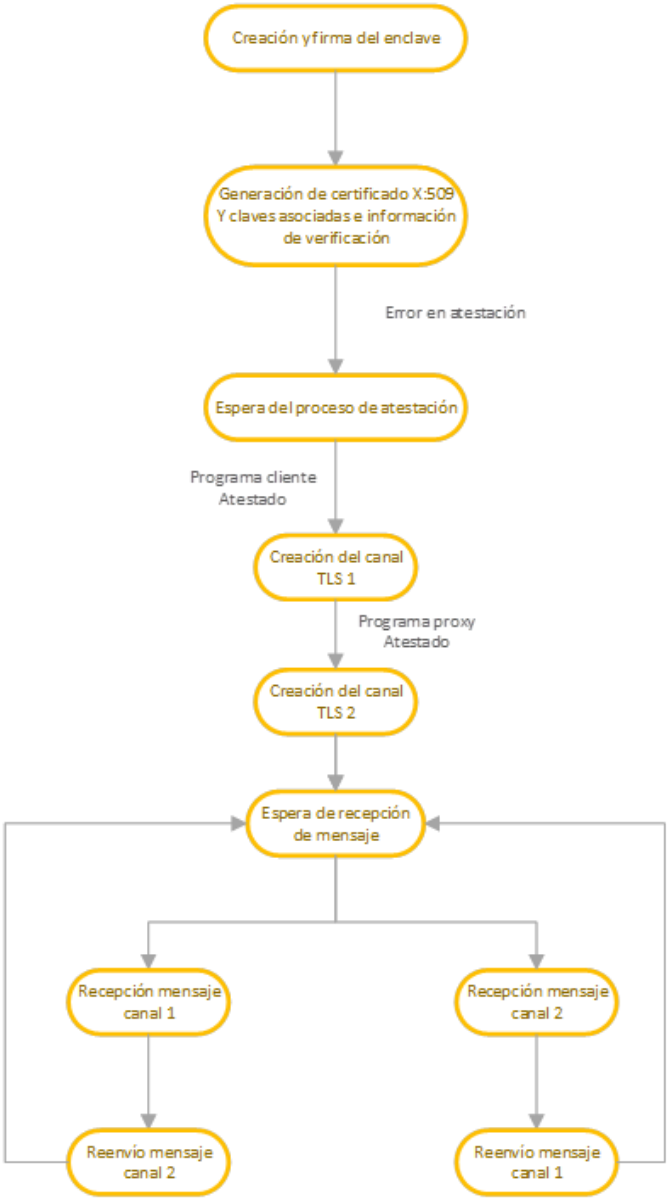


Figura 24. Diagrama de Estados del programa proxy DNS ejecutado en el enclave

Respecto al diagrama del programa del enclave, los primeros estados están relacionados con la creación de los elementos necesarios para la comunicación con agentes externos, como puede ser la generación de claves y firma del enclave, como con la creación del informe de verificación del servidor y generación de los certificados X.509 para la configuración y establecimiento del canal vía protocolo TLS.

Una vez creada la información y generada la variable con la configuración TLS, el programa pasará al estado de espera para que los programas externos realicen el proceso de atestación.

Por último, se pasará al estado espera a recepción de mensajes, que pasaría a el estado de reenvío de mensaje por el otro canal, es decir, por el que no ha llegado el mensaje. Esto se repetirá de manera infinita hasta que se cierre la comunicación socket.

9.2.2.3 Programa Proxy ejecutado fuera del enclave

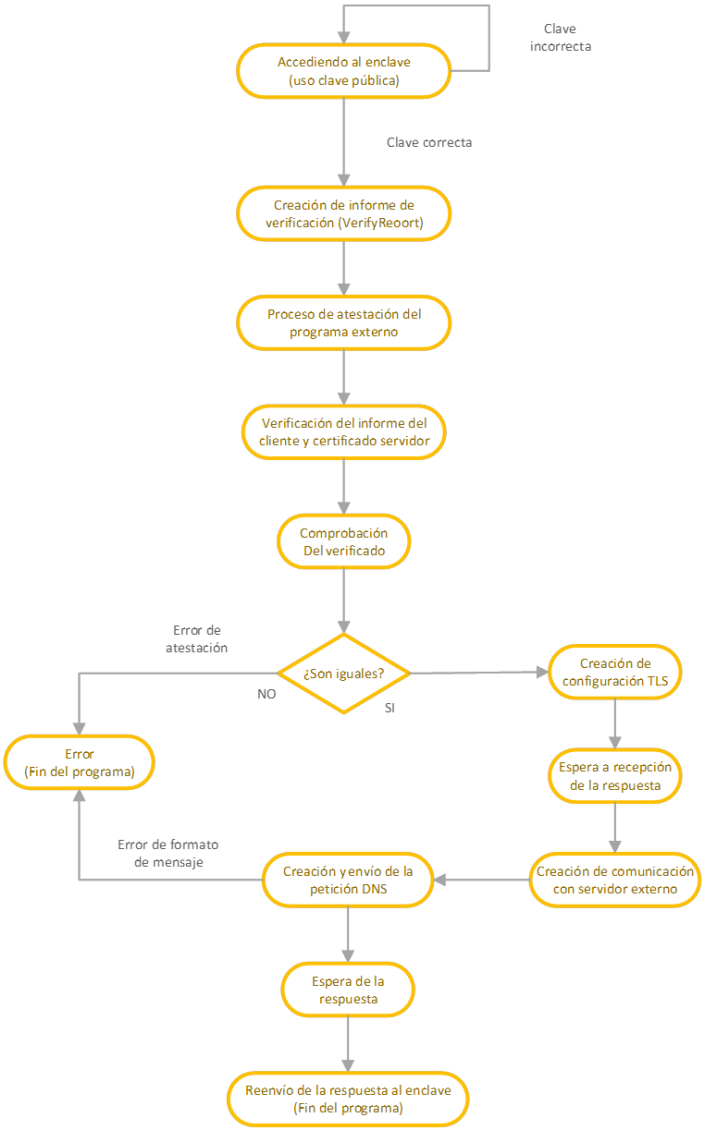


Figura 25: Diagrama de Estados del programa Proxy DNS ejecutado fuera del enclave

Como se puede observar en la Figura 25, la parte inicial del diagrama es idéntica a la del programa del cliente, esto es debido a que se hace uso de las mismas funciones utilizadas en el proceso de atestación.

La diferencia con el script del cliente reside en que, en este caso, tras crear la comunicación TLS con el enclave pasa al estado de espera a recibir el mensaje del enclave. Luego, una vez pasado por el estado de depuración del mensaje, donde se extrae la cadena de caracteres del flujo de bytes, se pasa a la creación de la comunicación con el servidor DNS de Google y se crea la petición DNS a enviar. En el caso de que el mensaje no esté bien creado, el programa pasará al estado de error, dando por finalizado el programa.

9.2.3 Diagrama de actividades

El diagrama de actividad, también conocido como diagrama de flujo, representa el flujo de trabajo de la solución a desarrollar.

Al igual que en el diagrama de estados, estos diagramas también se dividirán en base a los tres scripts que componen la solución. El objetivo es representar los diferentes caminos que toman los programas en base a los outputs de información que van ofreciendo los programas.

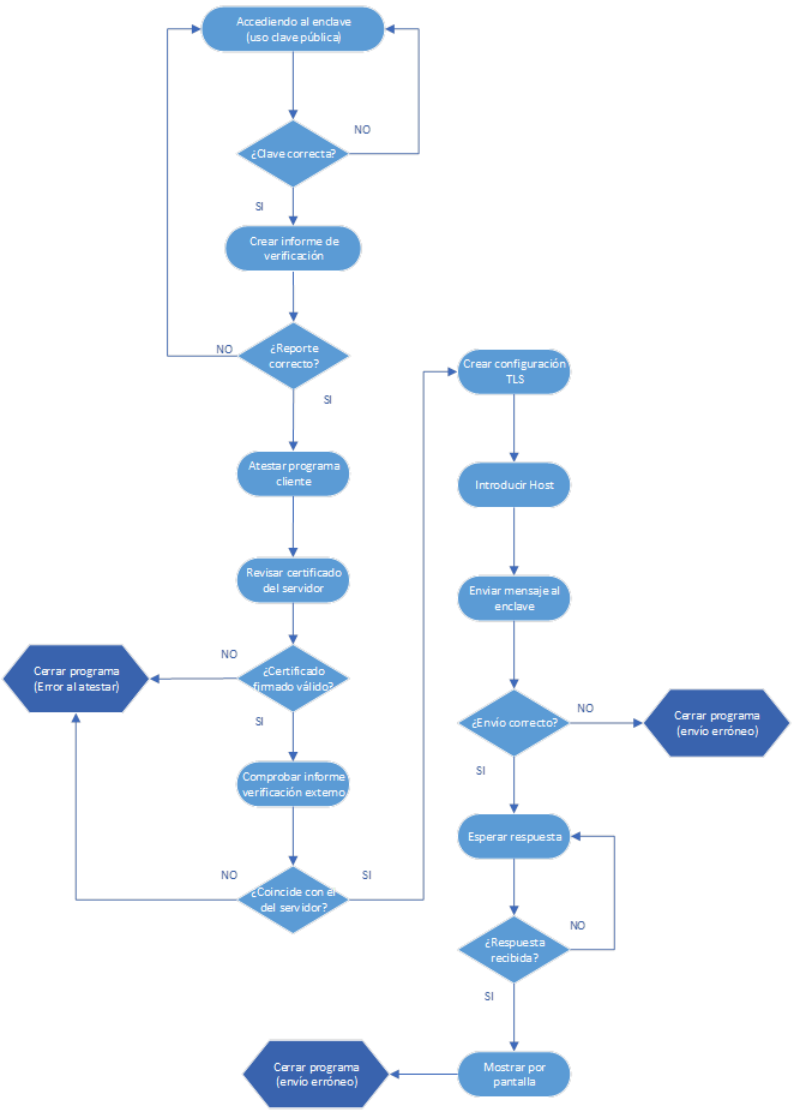


Figura 26: Diagrama de flujo del programa cliente

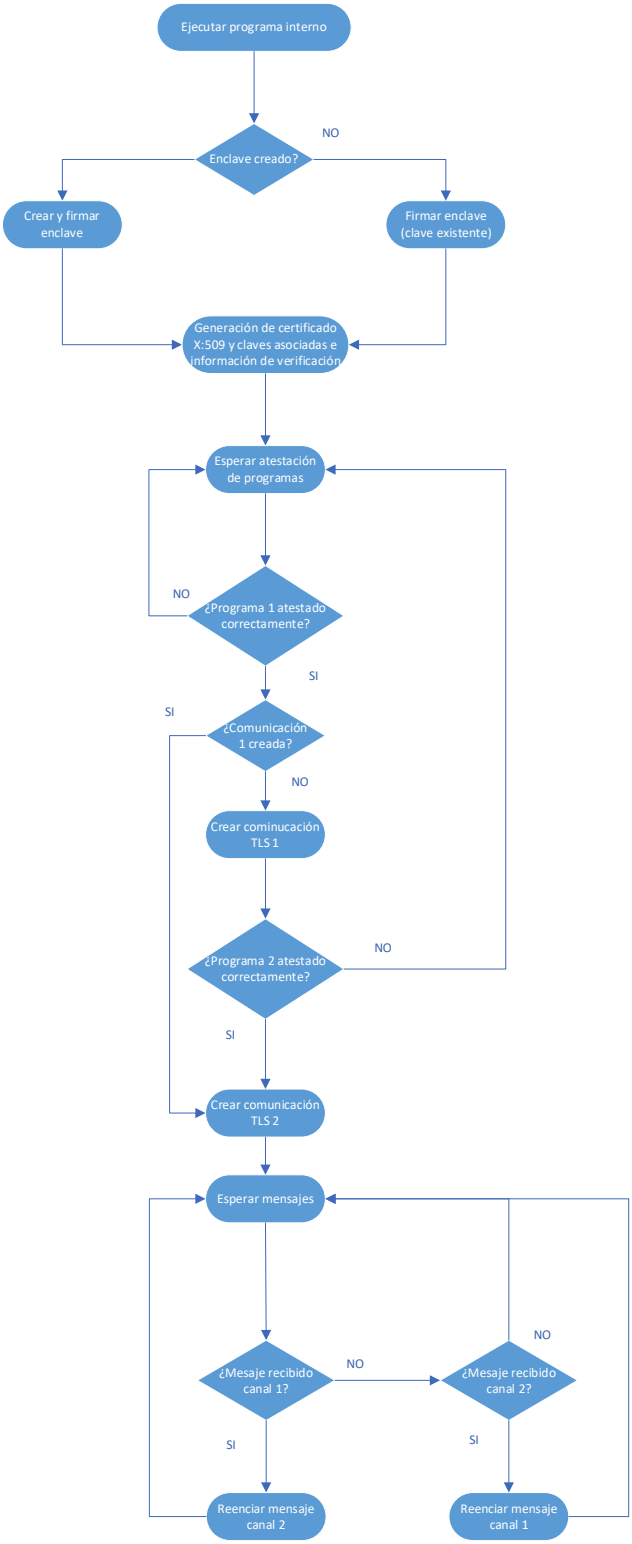


Figura 27: Diagrama de Flujo del programa Proxy DNS ejecutado dentro del enclave

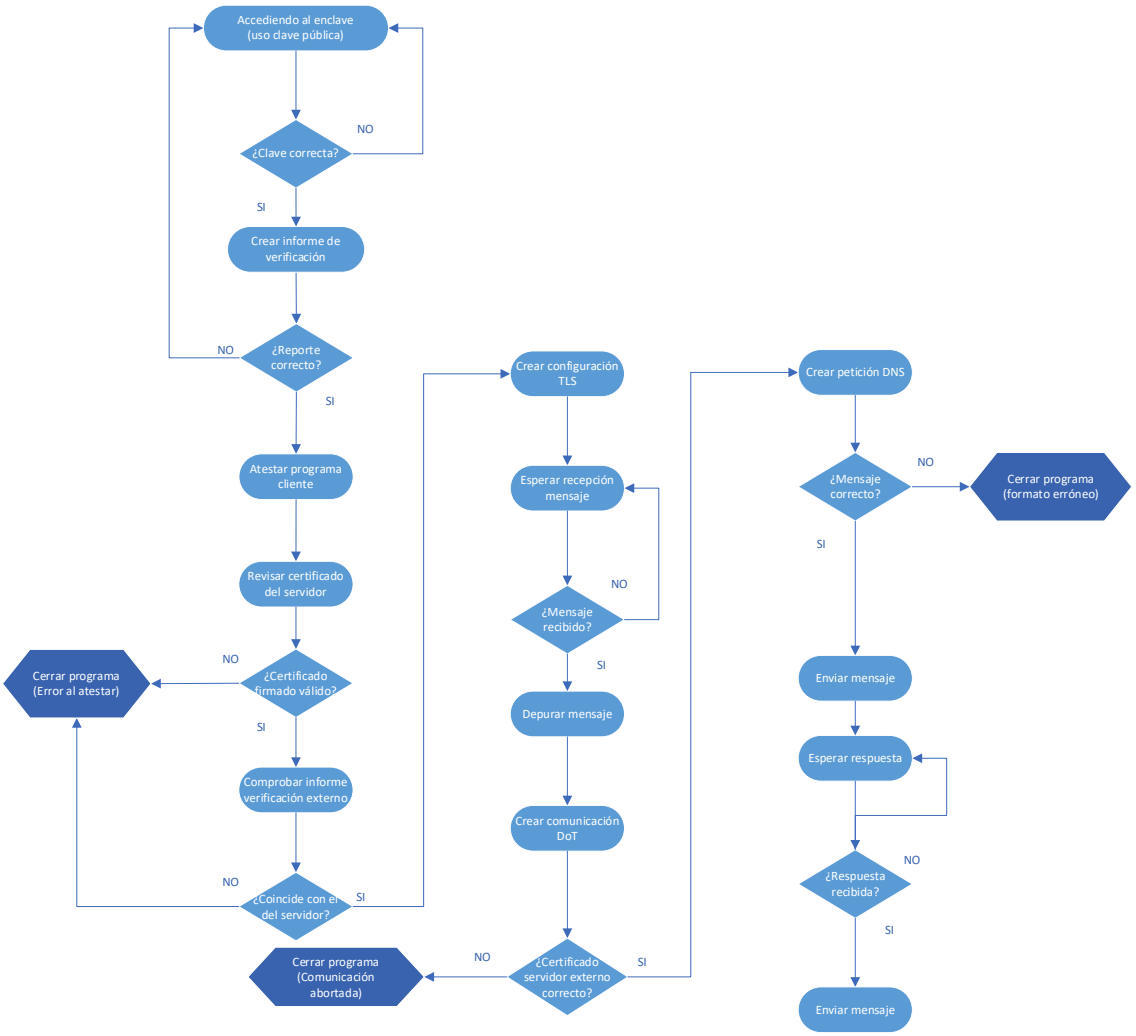


Figura 28: Diagrama de Flujo del programa Proxy DNS ejecutado fuera del enclave

Como se puede observar, tienen una apariencia similar a los diagramas de estado. No obstante, en este caso, en función de si la información obtenida del proceso anterior es correcta o no, el programa realiza una acción u otra.

9.3 Diseño de bajo nivel

En este subapartado, se realiza el diseño de bajo nivel que tiene la solución que se ha desarrollado para el presente trabajo. Dicho diseño se centra en analizar con mayor profundidad el diseño de alto nivel. Para ello, se realizará el diseño de bajo nivel en función de las fases que tiene la solución y no en base al bloque en el que está dividido el mismo. Además, en este punto se hará mención de las llamadas al enclave que se están realizando.

9.3.1 Creación ejecución y acceso al enclave seguro

El primer paso es la creación del enclave. Como se ha comentado previamente, el enclave se crea después de compilar el programa que se va a ejecutar dentro del enclave. Esto se realiza

con el comando ego sign server el cual realiza la creación del enclave, el certificado X.509 del enclave y el par de claves privada y pública para poder acceder a enclave. En este punto se realiza la instrucción **ecreate** que sirve para crear el enclave.

Una vez creado el enclave, el siguiente paso es ejecutar los ficheros que interactuarán con los enclaves y, previamente, ejecutar el servidor para abrir el enclave. Con el código del enclave en ejecución, se ejecutan los siguientes comandos en terminales diferentes:

- **ego run enclave:** para ejecutar el código dentro del enclave.
- **./client -s `ego signerid public.pem`**: se ejecuta el programa del cliente firmado con la clave pública generada al crear el enclave.
- **./proxy -s `ego signerid public.pem`**: se ejecuta el programa del proxy DNS firmado con la clave pública generada al crear el enclave.

En primer lugar, se ejecuta el código que correrá dentro del enclave. De esta forma, se llamará a la instrucción **einit** para inicializar el enclave**.** Al inicializar el enclave, este se quedará a la espera de que programas externos al enclave realicen el proceso de atestación para ver si sin fiables o no.

Una vez ejecutado el servidor, se realiza la creación de las configuraciones TLS para que, tanto el programa cliente como el programa del servidor proxy, puedan realizar el proceso de atestación. Como se puede observar en la Figura 29, en primer lugar, se crea una plantilla de un certificado X.509, introduciendo el número serie, el nombre común y la fecha de expiración del certificado.

```
func CreateAttestationServerTLSConfig(getRemoteReport func([]byte) ([]byte, error)) (*tls.Config, error) {
	serialNumberLimit := new(big.Int).Lsh(big.NewInt(1), 128)
	serialNumber, err := rand.Int(rand.Reader, serialNumberLimit)
	if err != nil {
		return nil, err
	}

	template := &x509.Certificate{
		SerialNumber: serialNumber,
		Subject:      pkix.Name{CommonName: "EGo"},
		NotAfter:     time.Now().AddDate(1, 0, 0),
	}

	priv, err := ecdsa.GenerateKey(elliptic.P256(), rand.Reader)
	if err != nil {
		return nil, err
	}

	cert, err := CreateAttestationCertificate(getRemoteReport, template, template, &priv.PublicKey, priv)
	if err != nil {
		return nil, err
	}

	return &tls.Config{
		Certificates: []tls.Certificate{
			{
				Certificate: [][]byte{cert},
				PrivateKey:  priv,
			},
		},
	}, nil
}
```

Figura 29: Configuración TLS para proceso de atestación

El siguiente paso es la creación de un par de claves pública y privada asociadas al certificado del servidor mediante la función de la librería "crypto/ecdsa": ***priv, err := ecdsa.GenerateKey(elliptic.P256(), rand.Reader).*** El certificado creado no está firmado por

ninguna CA, es decir, que está autofirmado por el servidor interno del enclave. Por último, se llama a la función ***CreateAttestationCertificate()*** las siguientes variables:

- ***getRemoteReport()*** una función que crear un reporte de verificación remoto de bytes para el proceso de atestación. Este reporte se basa en la realización del hash a la clave pública asociada a la clave privada que se ha generado con la función GenerateKey(). La función está indicada en la Figura 31.
- ***template***: es la plantilla del certificado X.509 a crear.
- ***template***: es la plantilla del certificado X.509 que servirá para autofirmar el certificado a crear.
- ***&priv.PublicKey***: la clave publica generada a partir de la clave privada.
- ***Priv***: la clave privada creada previamente.

Una vez creado el certificado X.509, este se devuelve al programa del enclave para poder inicializar una comunicación TLS para el proceso de atestado. Cabe destacar que, en la Figura 30, se muestra el proceso de creación del certificado del servidor.

```
func CreateAttestationCertificate(getRemoteReport func([]byte) ([]byte, error), template, parent *x509.Certificate, pub, priv interface{})
([]byte, error) {
        // get report for the public key
        hash, err := hashPublicKey(pub)
        if err != nil {
                return nil, err
        }
        report, err := getRemoteReport(hash)
        if err != nil {
                return nil, err
        }

        template.ExtraExtensions = append(template.ExtraExtensions, pkix.Extension{Id: oidOeNewQuote, Value: report})

        return x509.CreateCertificate(rand.Reader, template, parent, pub, priv)
}
```

Figura 30: Función de creación del certificado del servidor interno del enclave

```
func GetRemoteReport(reportData []byte) ([]byte, error) {
        if len(reportData) > maxReportData {
                return nil, errReportDataTooLarge
        }

        var report *C.uint8_t
        var reportSize C.size_t

        res, _, errno := syscall.Syscall6(
                sysGetRemoteReport,
                uintptr(unsafe.Pointer(&reportData[0])),
                uintptr(len(reportData)),
                0,
                0,
                uintptr(unsafe.Pointer(&report)),
                uintptr(unsafe.Pointer(&reportSize)),
        )
        if err := oeError(errno, res); err != nil {
                return nil, err
        }

        result := C.GoBytes(unsafe.Pointer(report), C.int(reportSize))
        syscall.Syscall(sysFreeReport, uintptr(unsafe.Pointer(report)), 0, 0)
        return result, nil
}
```

Figura 31: Función para la generación del reporte firmado por el enclave para el proceso de atestación

9.3.2 Proceso de atestación y acceso al enclave seguro

Esta parte es común tanto para el programa cliente como para el programa del proxy que se ejecuta fuera del enclave. lo primero que se realiza es, mediante la función ***flag.String("s", "", "signer ID")***, coger de la terminal lo que se ha introducido a través de la opción **-s** en el comando de ejecución del programa que, en este caso, es la clave pública de acceso al enclave y la guarda en la variable ***signerArg***. Luego, con la función ***signer, err = hex.DecodeString(*signerArg)*** traduce la clave pública que está en hexadecimal y la traduce a un conjunto de bytes y los almacena en la variable **signer**. Además, en el caso de que la variable esté a cero o haya saltado un error, la siguiente parte del código representaría el error por pantalla. La finalidad de estas líneas de código es inicializar el programa y cumplir con el primer paso para poder acceder al enclave.

```
if err != nil {
                panic(err)
        }
        if len(signer) == 0 {
                flag.Usage()
                return
        }
```

El siguiente paso es atestar el cliente en el servidor que se está ejecutando dentro del enclave para permitir a este realizar llamadas **ecalls** al enclave y poder acceder a la información que está manejando en el enclave.

Lo primero que se realiza, es la configuración del objeto de configuración TLS para la comunicación con el servidor que se ejecuta dentro del enclave seguro. Para ello se realiza la llamada a ***eclient.CreateAttestationClientTLSConfig(verifyReport).*** Esta función es propia de la librería de enclaves ego y sirve para que programas externos al puedan acceder al enclave[22]. Esta llamada recibe como parámetro el reporte de verificación del cliente realizado en una función aparte. El reporte de verificación consiste en comprobar que los valores de configuración utilizados para el enclave coincidan con los que está introduciendo el cliente.

```
func verifyReport(report attestation.Report) error {
        if report.SecurityVersion < 2 {
                return errors.New("invalid security version")
        }
        if binary.LittleEndian.Uint16(report.ProductID) != 1234 {
                return errors.New("invalid product")
        }
        if !bytes.Equal(report.SignerID, signer) {
                return errors.New("invalid signer")
        }

        return nil
}
```

Figura 32: Función para crear el reporte de verificación del programa externo

La función ***CreateAttestationClientTLSConfig()*** está definida dentro del fichero de configuración eclient que, como se muestra en la Figura 33, añade la información del reporte creado del programa e introduce los valores del reporte dentro de opciones de configuración a revisar en

el proceso de atestación. Cabe destacar que, para la verificación y realización del proceso de atestación se realiza la llamada a la función de ***CreateAttestationClientTLSConfig*** que se muestra en la Figura 34.

```
func CreateAttestationClientTLSConfig(verifyReport func(attestation.Report) error, opts ...AttestOption) *tls.Config {
	var appliedOpts internal.Options
	for _, o := range opts {
		o.apply(&appliedOpts)
	}

	return internal.CreateAttestationClientTLSConfig(
		verifyRemoteReport,
		appliedOpts,
		func(rep internal.Report) error { return verifyReport(attestation.Report(rep)) },
	)
}
```

Figura 33: Función para revisar el reporte de verificación del cliente

Como se puede observar, al entrar en la función ***CreateAttestationClientTLSConfig(),*** lo primero que se realiza es comprobar el contenido del certificado del servidor. Una vez comprobado que el certificado del servidor no está vacío, el siguiente paso es comprobar la entidad que ha firmado el certificado mediante la función de la Figura 30, ***cert.Verify(x509.VerifyOptions{Roots: roots})*** que, en este caso, comprueba que el certificado esté autofirmado. Para la comprobación de si el certificado se ha firmado por una entidad válida, se realiza el hash de la clave pública del certificado y se comprueba con la variable ***report.Data***, que contiene el hash de la clave pública generada en base a la clave privada del servidor TLS.

La comprobación se realiza con la función ***bytes.Equal***, la cual compra byte a byte el contenido de los dos hashes. Si no son iguales daría un mensaje de error. En el caso opuesto, la función devolvería una variable ***tls.Config*** con los valores necesarios para crear una comunicación entre el cliente externo y el enclave a través de una comunicación TLS.

```
func CreateAttestationClientTLSConfig(verifyRemoteReport func([]byte) (Report, error), opts Options, verifyReport func(Report) error)
*tls.Config {
	verify := func(rawCerts [][]byte, verifiedChains [][]*x509.Certificate) error {
		// parse certificate
		if len(rawCerts) <= 0 {
			return errors.New("rawCerts is empty")
		}
		cert, err := x509.ParseCertificate(rawCerts[0])
		if err != nil {
			return err
		}

		// verify self-signed certificate
		roots := x509.NewCertPool()
		roots.AddCert(cert)
		_, err = cert.Verify(x509.VerifyOptions{Roots: roots})
		if err != nil {
			return err
		}

		// verify embedded report
		for _, ex := range cert.Extensions {
			if ex.Id.Equal(oidOeNewQuote) {
				report, err := verifyRemoteReport(ex.Value)
				if err != nil && err != opts.IgnoreErr {
					return err
				}
				if !bytes.Equal(report.Data[:len(hash)], hash) {
					return errors.New("certificate hash does not match report data")
				}
				return verifyReport(report)
			}
		}

		return errors.New("certificate does not contain attestation report")
	}

	return &tls.Config{VerifyPeerCertificate: verify, InsecureSkipVerify: true}
}
```

Figura 34: Código para el proceso de atestación de un programa externo

Cabe destacar que, la función que se encarga de realizar la verificación del informe remoto creado por el cliente es la función que se muestra en la Figura 32, la cual comprueba el contenido del informe creado por el programa externo y, en caso de que sea correcto, devuelve un objeto ***attestation.Report*** con la información de autenticidad del informe.

```
func VerifyRemoteReport(reportBytes []byte) (attestation.Report, error) {
        if len(reportBytes) <= 0 {
                return attestation.Report{}, attestation.ErrEmptyReport
        }

        var claims, claimsLength uintptr

        res, _, errno := syscall.Syscall6(
                sysVerifyEvidence,
                uintptr(unsafe.Pointer(&reportBytes[0])),
                uintptr(len(reportBytes)),
                uintptr(unsafe.Pointer(&claims)),
                uintptr(unsafe.Pointer(&claimsLength)),
                0, 0,
        )

        var verifyErr error
        if err := oeError(errno, res); err != nil {
                if err.Error() != "OE_TCB_LEVEL_INVALID" {
                        return attestation.Report{}, err
                }
                verifyErr = attestation.ErrTCBLevelInvalid
        }

        defer syscall.Syscall(sysFreeClaims, claims, claimsLength, 0)

        report, err := internal.ParseClaims(claims, claimsLength)
        if err != nil {
                return attestation.Report{}, err
        }
        return attestation.Report{
                Data:            report.Data,
                SecurityVersion: report.SecurityVersion,
                Debug:           report.Debug,
                UniqueID:        report.UniqueID,
                SignerID:        report.SignerID,
                ProductID:       report.ProductID,
                TCBStatus:       report.TCBStatus,
        }, verifyErr
}
```

Figura 35: Función para comprobar el informe de verificación del cliente

En este punto, el programa externo quedaría atestado con el enclave y tanto el servidor interno como el cliente externo crearían una comunicación vía socket sobre el protocolo TLS para intercambiar información.

9.3.3 Transmisión de información a través del enclave

Una vez realizado el proceso de atestación por parte del programa cliente y el programa del proxy externo al enclave, el siguiente paso consiste en abrir la comunicación socket entre el enclave y ambos programas. Para ello, el programa ejecutado dentro del enclave crea dos sockets de forma simultánea y se queda escuchando a que los otros dos programas se conecten.

La función realizada se puede ver en el siguiente cuadro, donde se utiliza la configuración TLS obtenida con la siguiente llamada de la función vista anteriormente: (***enclave.CreateAttestationServerTLSConfig()***). En este caso, se abren dos sockets utilizando la IP localhost o 127.0.0.1 y habilitando los puertos 3080 y 3081, uno para cada comunicación. Como se puede observar, se utiliza una configuración TLS diferente para cada programa a comunicar con el enclave.

```
listener1, err := tls.Listen("tcp", "127.0.0.1:3080", tlsCfg1)
        if err != nil {
                panic(err)
        }
        defer listener1.Close()

        fmt.Println("listening socket 1...")

        listener2, err := tls.Listen("tcp", "127.0.0.1:3081", tlsCfg2)
        if err != nil {
                panic(err)
        }
        defer listener2.Close()
```

Ambos programas externos se conectan al socket de la misma forma, es decir, haciendo uso de la función ***conn, err := tls.Dial("tcp", *serverAddr, tlsConfig)***, la cual sirve para establecer una conexión TLS remota. Como se puede observar en el siguiente cuadro, la comunicación utiliza la configuración TLS recibida en el proceso de atestación, ya que en ella se encuentran los parámetros válidos para que un programa externo pueda realizar una comunicación con el enclave. Esto se realiza tanto en el programa cliente como en el proxy, cada uno de ellos con su ***tls.Config*** particular.

Con esto, todos los programas quedarían unidos, tal y como representa la Figura 17. Lo siguiente a realizar es que el usuario introduzca a través del teclado el nombre de dominio a realizar la petición y lo envía vía socket al enclave mediante la función ***conn.Write([]byte(input))***, la cual manda un flujo de bytes que está compuesto por el casteo a bytes de la cadena que contiene el nombre de dominio. En este punto se realiza una llamada al enclave **ecall**, la cual sirve para enviar la información del nombre de dominio al enclave.

El programa del enclave, al recibir un mensaje a través de la comunicación 1, realiza la función ***go handleConnection(conn1, conn2)***, la cual tiene como finalidad la retransmisión de la información que llega por uno de las comunicación socket a la otra. La Figura 36 muestra la composición de la función **handleConnection**.

```
func handleConnection(srcConn, destConn net.Conn) {
        defer srcConn.Close()

        buffer := make([]byte, 1024)
        for {
                n, err := srcConn.Read(buffer)
                if err != nil {
                        fmt.Println(err)
                        return
                }

                receivedData := buffer[:n]
                fmt.Printf("Received from %s: %s\n", srcConn.RemoteAddr(), receivedData)

                _, err = destConn.Write(receivedData)
                if err != nil {
                        fmt.Println(err)
                        return
                }
        }
}
```

Figura 36: Función para retransmisión de mensajes que llegan al enclave

9.3.4 Creación de comunicación DoT y transmisión de información

Antes la creación del mensaje de petición DNS, es necesario depurar la información que el servidor proxy recibe del enclave, ya que la información viene en formato bytes mediante la función ***conn.Read(response)*** Para ello se ha hecho uso de las siguiente tres funciones:

- removeNullBytes().
- bytesToStrings().
- hexToASCII().

En primer lugar, se utiliza la función ***removeNullBytes(response)*** para quitar los caracteres nulos que puedan quedar en el mensaje recibido. Esto es debido a que el buffer de transmisión se ha configurado a 512 bytes por lo que, si el mensaje a enviar tiene un valor inferior a este, el resto de huecos se autocompletarán con el carácter nulo. En segundo lugar se realiza la función ***bytesToStrings(input []byte))***, la cual realiza un casteo de para traducir el flujo de bytes en caracteres de una cadena hexadecimal. Por último, se utiliza la función ***hexToASCII(exStr string)*** para convertir el código hexadecimal en el código ASCII con el valor de las letras correspondientes al mensaje que se ha mandado desde el cliente.

Todo este proceso es necesario, ya que el repositorio **miekg/DNS** utilizado para la creación del servidor proxy recibe una cadena de caracteres en un formato específico (example.com.) para crear la consulta DNS.

Una vez depurada la información recibida en el proxy, el siguiente paso es la creación de la conexión DNS entre el servidor proxy y el servidor de Google 8.8.8.8. Para ello, mediante el código indicado en el siguiente cuadro, se configura una conexión DNS sobre el protocolo TLS con el servidor.

```
tlsConfig2 := &tls.Config{
        ServerName: "dns.google",
}
dnsClient := &dns.Client{
        Net:      "tcp-tls",
        TLSConfig: tlsConfig2,
}
```

Como se puede observar, se indica el servidor al que se va a acceder y, posteriormente, se configura el cliente mediante la estructura que muestra la Figura 37. No se utiliza el segundo parámetro, porque no se realiza la conexión a través del protocolo UDP.

```
type Client struct {
        Net       string      // if "tcp" or "tcp-tls" (DNS over TLS) a TCP query will be initiated, otherwise an UDP one (default is "" for UDP)
        UDPSize   uint16      // minimum receive buffer for UDP messages
        TLSConfig *tls.Config // TLS connection configuration
```

Figura 37: Estructura para la creación del cliente DNS

Una vez creada la conexión, el siguiente paso a realizar es la creación de la petición DNS y enviarla al servidor externo. La creación del mensaje se realiza mediante la llamada a la función ***SetQuestion(resultString, dns.TypeA)*** a la cual se le pasan los parámetros que componen el nombre DNS a traducir y el tipo de consulta DNS que se va a realizar.

Por último, el envío del mensaje se realiza a través de la llamada a la función ***dnsClient.Exchange(query, "8.8.8.8:853")***, en la que se le pasa el mensaje a enviar y la dirección IP y puerto al que se va a conectar. En este caso, se utiliza la IP del DNS de Google y se utiliza el puerto 853, el cual es un puerto **well-known** reservado para DoT. LA función anterior realiza una consulta síncrona en la que el proxy envía el mensaje y espera a la respuesta.

Por último, el proxy reenvía el flujo de bytes recibidos de la respuesta DNS al enclave vía comunicación socket haciendo uso de la función ***conn.Write([]byte(input)).***

9.3.5 Visualización del resultado

Cuando la respuesta DNS llega al programa cliente a través del enclave, este la muestra por pantalla haciendo uso de la llamada a la función ***Unpack(response[:n])***. El "**:n**" sirve para coger únicamente los bytes no nulos. Luego, haciendo uso de la función Println, ***fmt.Println(responseMsg.String())***, se muestra el resultado en la terminal en la que se está ejecutando el programa cliente.

10 DESCRIPCIÓN DE LOS RESULTADOS

En este apartado se evalúan y analizan los resultados que se han obtenido a la hora de ejecutar la solución propuesta que se ha descrito en el apartado anterior. El objetivo de este apartado es mostrar el funcionamiento de la solución desplegada. Se van a describir y analizar dos tipos de resultados que se han obtenido a la hora de realizar las pruebas de la solución descrita previamente.

Por un lado, se describien los resultados de la ejecución de la solución realizada para analizar las funcionalidades de la solución. En estas pruebas servirán para comprobar que la solución planteada cumple con el objetivo principal planteado en el presente trabajo.

Por otro lado, se realizarán una serie de pruebas que permitan analizar el rendimiento a nivel de seguridad que ofrece la solución. Las pruebas a realizar consisten en comprobar el difícil acceso que tienen los programas externos a los enclaves y, por otro lado, se comprobará que el programa cliente que accede al enclave no puede conectarse a otro servidor Proxy DNS sospechoso.

10.1 Evaluación de la Solución

Como se ha comentado previamente, las primeras pruebas realizadas son pruebas de comportamiento del programa. Las pruebas de funcionamiento servirán para medir la utilidad del programa y de los enclaves. Tras realizar la ejecución del programa, las siguientes imágenes muestran el resultado obtenido del programa.

En primer lugar, se ve como el programa que manejaría el usuario realiza el proceso de atestación correctamente al enclave y se conecta al servidor interno para el intercambio de información. Luego, al recibir los bytes de la respuesta DNS del nombre de dominio indicado por teclado (Mostrado en la Figura 38) se muestra por pantalla el resultado de la consulta DNS indicando el tipo de consulta que se ha realizado.

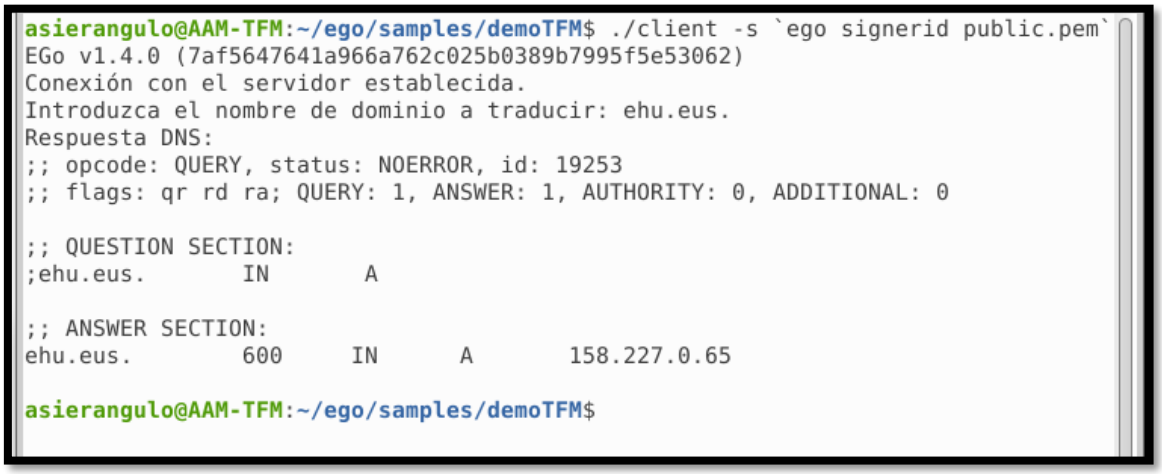


Figura 38: resultado de la ejecución del programa cliente

Si se observa el resultado del programa que se está ejecutando dentro del entorno seguro, lo primero que se aprecia es la generación de la clave pública para el acceso al enclave, clave que utilizan los programas que desean acceder al enclave.

Luego, se observa cómo se accede al enclave y se crear las dos comunicaciones seguras para conectarse tanto al proxy DNS como al programa del usuario. Al ver la Figura 39, se puede observar que, cuando el enclave recibe un mensaje, muestra por pantalla la información del extremo que ha realizado dicho envío. Esto sirve para llevar un registro de si la información entra al enclave o no.

Por último, aparece dos veces el mismo mensaje: E0F. Esto indica que el otro extremo ha finalizado la comunicación. Sin embargo, el programa interno del enclave se queda ejecutando constantemente esperando a que más clientes puedan conectarse al enclave y seguir resolviendo consultas DNS.

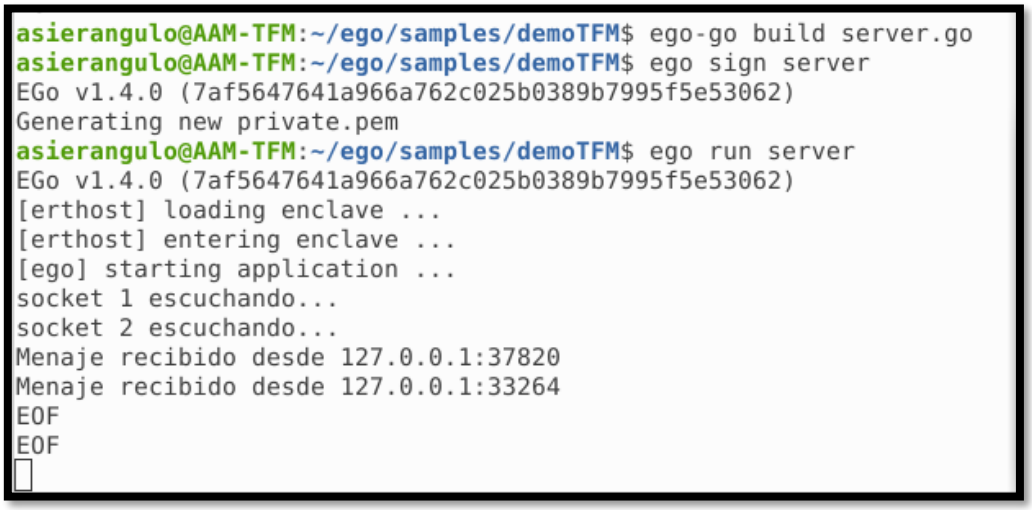


Figura 39: Resultado de la ejecución del programa interno del enclave

Por último, si se observa el script del proxy DNS, se ve como, al igual que el programa del cliente, realiza de manera exitosa el proceso de atestación del enclave y se queda a la espera de recibir el mensaje enviado desde el interior del enclave.

Tras recibir el flujo de bytes, se realiza la depuración de la información recibida hasta dejar la información como la cadena de caracteres que envió el cliente, tal y como se puede apreciar en la Figura 40. Por último, una vez que realiza la consulta con el servidor externo, este envía la respuesta al enclave de forma transparente y, posteriormente, se cierra el programa, haciendo que en el programa interno del enclave se visualicen los caracteres comentado previamente.

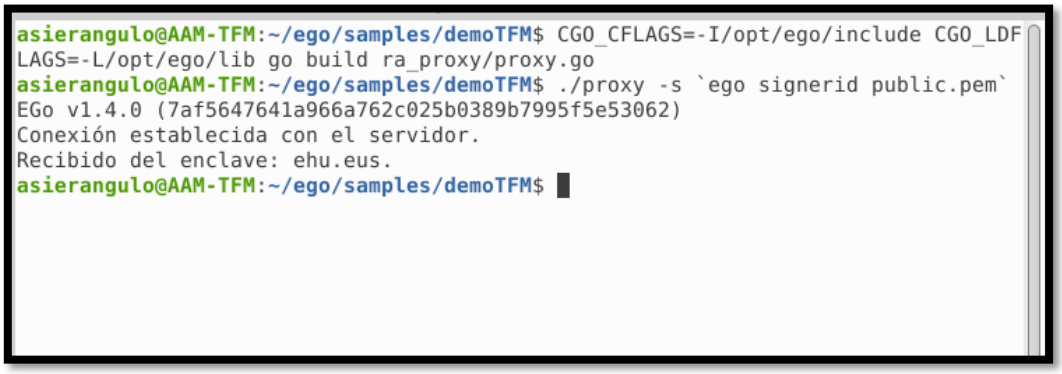


Figura 40: Resultado de la ejecución del proxy DNS

Una vez ejecutados los programas, se puede concluir que el desarrollo planteado realiza una consulta DNS a través de enclavas seguros. Además, mediante el comando "**dig**", se realiza la verificación de la respuesta DNS para ver si el resultado obtenido es correcto o no. Como se puede observar en la Figura 41, se obtiene la misma dirección IP que con la ejecución de la solución desarrollada. Por lo que se pude concluir diciendo que el programa funciona correctamente.

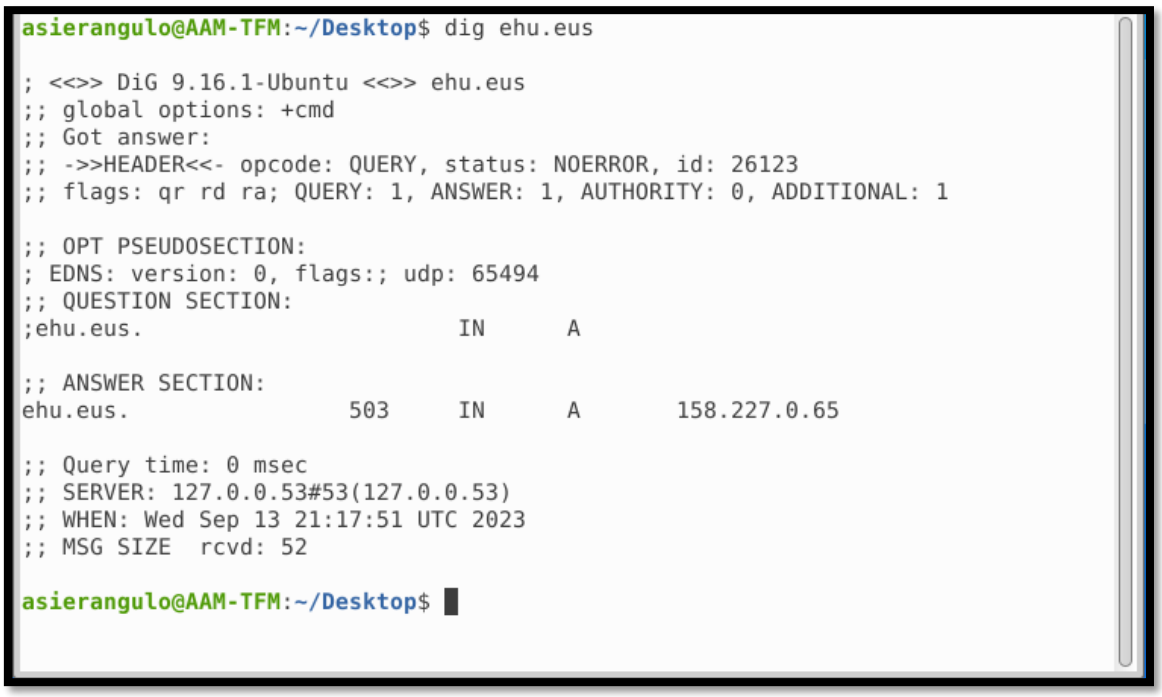


Figura 41: Resultado del comando dig

10.2 Evaluación de resultados de seguridad

Una vez asegurado que el programa funciona correctamente, la segunda parte de la descripción de los resultados consiste en comprobar los mecanismos de seguridad que ofrece el desarrollo de la solución realizada.

10.2.1 Evaluación de consulta DNS externa

La primera prueba que se han realizado han sido la comprobación del uso del protocolo TLS para la comunicación entre el proxy DNS y el servidor DNS de Google. Esta prueba se realiza para asegurar que la comunicación se dota de mecanismos de confidencialidad e integridad.

Debido a que en la máquina virtual de Microsoft Azure no se puede ejecutar el programa wireshark, se ha hecho del comando **tshark** para poder realizar la captura de los paquetes intercambiados en la interfaz **eth0** que es la interfaz con la que la máquina virtual sale a Internet. El comando utilizado ha sido el siguiente: ***tshark -i eth0.*** Una vez ejecutado el comando y los programas de la solución, se ha obtenido el siguiente resultado:

```
 1 0.000000000    10.0.0.4 → 8.8.8.8      TCP 74 46184 → 853 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1793440170 TSecr=0 WS=128
 2 0.002332513     8.8.8.8 → 10.0.0.4     TCP 74 853 → 46184 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 TSval=1818294238 TSecr=1793440170 WS=256
 3 0.002367613    10.0.0.4 → 8.8.8.8      TCP 66 46184 → 853 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1793440172 TSecr=1818294238
 4 0.002487214    10.0.0.4 → 8.8.8.8      TLSv1 328 Client Hello
 5 0.005169528     8.8.8.8 → 10.0.0.4     TCP 66 853 → 46184 [ACK] Seq=1 Ack=263 Win=66816 Len=0 TSval=1818294240 TSecr=1793440173
 6 0.006900838     8.8.8.8 → 10.0.0.4     TLSv1.3 4878 Server Hello, Change Cipher Spec, Application Data
 7 0.006932538    10.0.0.4 → 8.8.8.8      TCP 66 46184 → 853 [ACK] Seq=263 Ack=4813 Win=62080 Len=0 TSval=1793440177 TSecr=1818294242
 8 0.018844303    10.0.0.4 → 8.8.8.8      TLSv1.3 130 Change Cipher Spec, Application Data
 9 0.019057504    10.0.0.4 → 8.8.8.8      TLSv1.3 115 Application Data
10 0.021752819     8.8.8.8 → 10.0.0.4     TCP 66 853 → 46184 [ACK] Seq=4813 Ack=376 Win=66816 Len=0 TSval=1818294257 TSecr=1793440189
11 0.025943441     8.8.8.8 → 10.0.0.4     TLSv1.3 131 Application Data
12 0.026084642    10.0.0.4 → 8.8.8.8      TLSv1.3 90 Application Data
13 0.026098342    10.0.0.4 → 8.8.8.8      TCP 66 46184 → 853 [FIN, ACK] Seq=400 Ack=4878 Win=64128 Len=0 TSval=1793440196 TSecr=1818294261
14 0.029596461     8.8.8.8 → 10.0.0.4     TCP 66 853 → 46184 [FIN, ACK] Seq=4878 Ack=401 Win=66816 Len=0 TSval=1818294265 TSecr=1793440196
15 0.029609761    10.0.0.4 → 8.8.8.8      TCP 66 46184 → 853 [ACK] Seq=401 Ack=4879 Win=64128 Len=0 TSval=1793440200 TSecr=1818294265
```

Figura 42: Captura tshark en la interfaz eth0

Si se observa la Figura 42, se puede apreciar cómo una vez realizada la fase de saludo y establecimiento de conexión, se negocian las características de cifrado que se van a utilizar en la comunicación. Una vez cifrada la comunicación, el proxy DNS realiza la petición DNS y espera a la respuesta.

Con esto, se verifica que la comunicación con el servidor público externo se realiza a través de una comunicación segura, ofreciendo los mecanismos de confidencialidad e integridad. A la comunicación.

10.2.2 Intento de Acceso de cliente externo sin clave pública

En esta segunda prueba se va a comprobar que ningún agente o programa externo es capaz de acceder al enclave sin haber realizado un proceso de atestación previo. La primera prueba que se realiza a realizar es la prueba de acceso sin disponer de la clave pública o mediante una clave pública que no está asociada a la clave privada del enclave.

Para la realización de la prueba se ha generado un nuevo para de claves privada y pública asociada del enclave y se ha mantenido la clave pública obsoleta para que un usuario externo intente acceder al enclave. Al intentar acceder al enclave al enclave con la clave pública obsoleta, se visualiza el siguiente resultado.

```
asierangulo@AAM-TFM:~/ego/samples/demoTFM$ ./client -s `ego signerid public.pem`
EGo v1.4.0 (7af5647641a966a762c025b0389b7995f5e53062)
2023-09-13T20:49:01+0000.104943Z [(H)ERROR] tid(0x7f9b6fe2eb80) | :OE_CRYPTO_ERR
OR [/ertbuild/3rdparty/openenclave/openenclave-src/common/crypto/openssl/key.c:o
e_public_key_read_pem:232]
2023-09-13T20:49:01+0000.104966Z [(H)ERROR] tid(0x7f9b6fe2eb80) | :OE_CRYPTO_ERR
OR [/ertbuild/3rdparty/openenclave/openenclave-src/common/sgx/report_helper.c:oe
_sgx_get_signer_id_from_public_key:38]
2023-09-13T20:49:01+0000.104968Z [(H)ERROR] tid(0x7f9b6fe2eb80) | :OE_CRYPTO_ERR
OR [/edgelessrt/src/ert/oesign/oesignerid.c:oesignerid:35]

panic: encoding/hex: invalid byte: U+0052 'R'

goroutine 1 [running]:
main.main()
        /home/asierangulo/ego/samples/demoTFM/ra_client/client.go:30 +0x61d
asierangulo@AAM-TFM:~/ego/samples/demoTFM$
```

Figura 43: Error de acceso al enclave por clave pública errónea

Como se puede observar, como es evidente, el programa finaliza con un error de la clave pública, ya que no está asociada a la clave privada que se está utilizando para realizar la firma del enclave.

10.2.3 Acceso de cliente externo al enclave con clave pública

En esta segunda parte se va a ir un paso más allá y se realiza la prueba del acceso de un programa o agente externo que ha conseguido la clave pública del enclave. El objetivo de esta prueba es comprobar que, a pesar de disponer de la clave pública del enclave, si no se realiza el proceso de atestación no es posible el acceso a la información y código del programa que se está ejecutando en el enclave. La siguiente figura muestran las salidas en pantalla de la ejecución del programa:

```
<asierangulo@AAM-TFM:~/ego/samples/demoTFM$./client -s `ego signerid public.pem`
EGo v1.4.0 (7af5647641a966a762c025b0389b7995f5e53062)
panic: invalid product

goroutine 1 [running]:
main.main()
        /home/asierangulo/ego/samples/demoTFM/ra_client/client.go:43 +0x5c8
asierangulo@AAM-TFM:~/ego/samples/demoTFM$ ^C
asierangulo@AAM-TFM:~/ego/samples/demoTFM$
```

Figura 44: Resultado de Acceso de Usuario con clave pública, pero sin proceso de atestación

Como se puede observar, a la hora de realizar el proceso de atestación para el acceso al enclave, da un error en el proceso de verificación del informe de verificación creado por el cliente. En este punto, el enclave le deniega el acceso y se finaliza el programa.

Respecto al resultado que se muestra en la terminal del programa ejecutado en el interior del enclave, se muestra lo siguiente:

```
asierangulo@AAM-TFM:~/ego/samples/demoTFM$ ego run server
EGo v1.4.0 (7af5647641a966a762c025b0389b7995f5e53062)
[erthost] loading enclave ...
[erthost] entering enclave ...
[ego] starting application ...
socket 1 escuchando...
socket 2 escuchando...
remote error: tls: bad certificate
EOF
^C
asierangulo@AAM-TFM:~/ego/samples/demoTFM$
```

Figura 45: Resultado en el enclave del intento de acceso de un cliente sospechoso

Como se puede observar el enclave detecta que un programa quiere acceder a él mediante un informe de verificación erróneo o fraudulento. En este punto, el enclave le deniega el acceso y devuelve el error que en este caso ha habido un error en la comparación de los certificados. La comparación de los certificados viene indicada en el diseño de bajo nivel incluido en el apartado 9 del presente proyecto.

10.2.4 Cliente atestado accediendo a servidor malicioso

La última prueba que se realiza es el intento de conexión del cliente a un proxy DNS malicioso o no fiable. Esta prueba simula la manipulación de la dirección IP y puerto de la salida DNS del cliente para redirigirle las consultas a un servidor DNS o un proxy DNS malicioso.

Con esta prueba se desea comprobar que, en el caso de que esto ocurra, el cliente no sería capaz de acceder a este servidor ya qué estaría intentando realizar el proceso de atestación comprobando el informe de verificación del cliente y el certificado del servidor.

```
asierangulo@AAM-TFM:~/ego/samples/demoTFM$ CGO_CFLAGS=-I/opt/ego/include CGO_LDF
LAGS=-L/opt/ego/lib go build ra_client/client.go
asierangulo@AAM-TFM:~/ego/samples/demoTFM$ ./client -s `ego signerid public.pem`
EGo v1.4.0 (7af5647641a966a762c025b0389b7995f5e53062)
panic: dial tcp 127.0.0.1:3085: connect: connection refused

goroutine 1 [running]:
main.main()
        /home/asierangulo/ego/samples/demoTFM/ra_client/client.go:43 +0x5c8
asierangulo@AAM-TFM:~/ego/samples/demoTFM$
```

Figura 46: Resultado de cliente accediendo al servidor malicioso

Como se puede observar en la captura, el cliente intenta acceder al enclave, pero a este se le redirige a la dirección que quiere el servidor que se estaría ejecutando fuera del enclave. Entonces al intentar realizar el proceso de atestación no se detecta ninguna comunicación con un enclave y cierra la conexión del cliente sin acceder al servidor maligno.

10.3 Resumen y conclusiones de las pruebas realizadas

Como resumen de los resultados obtenido se puede concluir que se cumple con el objetivo principal del presente proyecto que se trata de securizar el protocolo DNS mediante el uso de enclaves seguros.

Además, se ha comprobado que para el acceso al enclave, en este caso, tiene dos métodos de acceso que hay que cumplir, por un lado, disponer de la clave pública del enclave y, por otro lado, autenticar el usuario mediante un proceso de atestación en el que se comprueba el informe de verificación del cliente con el del servidor y se facilita al cliente el certificado X.509 del servidor interno del enclave.

Por último, también se ha verificado que el cliente que se conecta al servidor interno del enclave, debido al proceso de atestación que debe pasar, no puede conectarse a otro servidor externo aun conociendo su dirección IP y puerto.

Por lo que se puede concluir con este apartado es que la solución propuesta ofrece mecanismos de seguridad de integridad y confidencialidad, mecanismos de cifrado asimétrico y certificados X.509 y mecanismos de autenticación y autorización.

11 PLANIFICACIÓN

En este apartado se realiza la explicación de la planificación que se ha seguido para llevar a cabo el Trabajo de Fin de Máster.

11.1 Grupo de trabajo

EL grupo de trabajo del proyecto se puede ver en la siguiente tabla.

Nombre y Apellidos	Responsabilidad	Rol
Eduardo Jacob	Ingeniero Senior	Dirección y supervisión del proyecto
Asier Angulo	Ingeniero Junior	Desarrollador del proyecto

Tabla 10: Grupo de trabajo del proyecto

11.2 Fases del proyecto

Para la realización del proyecto, se tienen en cuenta 6 paquetes de trabajo distintos. Además, al final de cada paquete, se ha definido un hito para facilitar el control de tareas realizadas y ofrecer una mayor sencillez a la monitorización y gestión de fechas. A continuación, se realizará una breve descripción de las tareas planificadas para la realización del presente proyecto.

P.T.1 Análisis preliminar

El análisis preliminar tiene como finalidad la familiarización con el punto de partida de este trabajo que son las tecnologías capaces de crear Entornos de Ejecución Fiables, más concretamente con los enclaves seguros SGX de Intel. Además, también se centra en buscar escoger una tecnología que carezca de mecanismos de seguridad y, haciendo uso de estos enclaves seguros, securizar la tecnología. El siguiente listado muestra las tareas a realizar que pertenecen a este punto, indicando su duración en horas.

- **Definición del proyecto y el Alcance**: Identificar los objetivos principales y el alcance del presente proyecto. (10h).
- **Análisis de punto de partida, entornos TTE**: Empezar a buscar información y familiarizarse con los Entornos de Ejecución Fiables. (30h).
- **Análisis de securización de DNS:** Seleccionar un protocolo o tecnología que no implemente mecanismos de seguridad y analizar cómo mejorar la seguridad del protocolo haciendo uso de los enclaves SGX. (20h).

P.T.2 Análisis de tecnologías TTE

En este segundo bloque, se realiza un análisis exhaustivo acerca de los enclaves seguros y tecnologías similares, para cumplir con los objetivos del proyecto. En este apartado se realiza una comparación acerca de los mecanismos de seguridad que ofrecen los diferentes Entornos de Ejecución Fiables.

- **Estudio de las alternativas para crear TTEs**: En este apartado se realiza la búsqueda de información necesaria para utilizarla posteriormente en la realización del Estado del arte (60h).
- **Comparación de los mecanismos de seguridad que ofrecen las tecnologías**: En este punto se realiza una comparativa de las tecnologías existentes en base al estado del arte anterior para determinar cuál es la que ofrece un mayor grado de seguridad. (15h).
- **Análisis de despliegue a realizar**: En este punto se realiza un análisis de las diferentes librerías existentes para poder desplegar una solución. También se busca información acerca de las diferentes formas existentes de desplegar una aplicación mediante el uso de enclaves seguros. (25h).
- **Selección de elementos a utilizar para la definición de la solución**: Se realiza una selección de los elementos, librerías… para poder empezar a desplegar la solución final. (20h).

P.T.3 Diseño de Alto nivel

En este paquete de trabajo se empiezan a plantear el diseño final que va a tener la aplicación en base a la selección realizada en el bloque anterior. En este bloque se empieza a separar la solución en los distintos bloques que la van a componer y se empiezan a diseñar los diferentes diagramas UML utilizados para entender mejor la aplicación y su objetivo final.

- **Definición de los elementos del proyecto**: Esta tarea se centra en la definición de los diferentes bloques que se van a tener que realizar para poder desplegar de manera éxitos la solución final. (10h).
- **Diseño del programa cliente**: En esta tarea se realizan los diagramas UML de la parte del cliente que realiza la petición DNS. (10h).
- **Diseño del programa proxy DNS interno del enclave**: En esta tarea se identifican los elementos a proteger en la aplicación y se plantea el diseño para almacenar la información dentro del enclave seguro. (15h).
- **Diseño del programa proxy DNS externo al enclave**: Esta tarea sirve para realizar el diseño del proxy DNS e identificar las diferentes formas que tiene de realizar la consulta DNS aplicando mecanismos de seguridad. (15h).
- **Diseño de las comunicaciones entre programas**: En este caso, el objetivo de la tarea es definir la forma en la que se van a comunicar los tres bloques planteados y la información que van a intercambiar. (10 h).

P.T.4 Desarrollo de la solución

En este grupo de tareas se realiza la programación de la solución final en base a los diagramas diseñados en el bloque anterior. Estas tareas sirven para definir la forma que se van a conectar los diferentes programas y las funciones de librerías a utilizar para definir las llamadas al enclave que se tienen que realizar. Además, también se programará el proxy DNS por separado para posteriormente realizar pruebas únicamente de este servidor.

- **Desarrollo de la solución**: Se realiza un pequeño diseño de bajo nivel en el que se seleccionan las funciones de librería que se van a utilizar, y la información a intercambiar entre programas y el formato de la misma.
- **Despliegue de proxy DNS a través de TLS:** Esta tarea consta de la realización de un proxy DNS al que un cliente le realiza una petición para que este reenvíe la consulta a un servidor externo a través del protocolo TLS. (40h).
- **Programación del del cliente**: Esta tarea sirve para crear el programa que utilizará el cliente para hacer una consulta DNS y enviar el nombre de dominio a traducir a través del enclave. (25h).
- **Programación del programa interno del enclave**: Esta tarea sirve para crear un enclave seguro y añadir en él el programa que se encargará de crear las conexiones para los diferentes clientes. (60h).
- **Programación del proxy DNS externo al enclave**: En esta parte, se añadirá al proxy DNS realizado en la tarea anterior el código correspondiente a la conexión con el enclave. (15h).

P.T.5 Realización de Pruebas

El siguiente paquete de tareas se centra en la realización de las pruebas para la comprobación del correcto funcionamiento de la solución desplegada. Por un lado, se realizan pruebas de funcionamiento ordinarias con la intención de encontrar algún error de programación o alguna parte del código que no funciona como se deseara. Por otro lado, se realizarán pruebas de seguridad para comprobar los mecanismos de seguridad que ofrecen los enclaves seguros.

- **Verificación del funcionamiento del Proxy DNS sobre TLS**: Verificación del correcto funcionamiento de la consulta DNS sobre el protocolo DNS, haciendo uso de capturadores de paquetes para ver que realmente la carga útil de los paquetes TLS van cifrados. (10h).
- **Verificación del funcionamiento de cliente externo accediendo al enclave**: En este apartado, se comprueba que el programa cliente puede conectarse al enclave y realizar llamadas al mismo desde el exterior. (20h).
- **Verificación del correcto funcionamiento del diseño planteado**: Se ejecutan todas las partes de la solución de forma simultánea para comprobar que la solución obtenida es correcta. (30h).
- **Verificación de denegación de acceso a cliente no deseado**: Se realizan pruebas cambiando el script del cliente para verificar que no todos los programas tienen acceso al enclave. (20h).
- **Verificación de no acceso a un servidor ajeno al enclave:** Según la forma en la que está codificado el cliente, aquí se realizan pruebas para ver que, si e cambia el enclave por un proxy sospechoso, el cliente no realiza la comunicación. (20h).

P.T.6 Realización de la documentación

Este último paquete indica la escritura en el documento de “Trabajo de Fin de Máster” las actividades que se han ido realizando en las tareas de los paquetes anteriores.

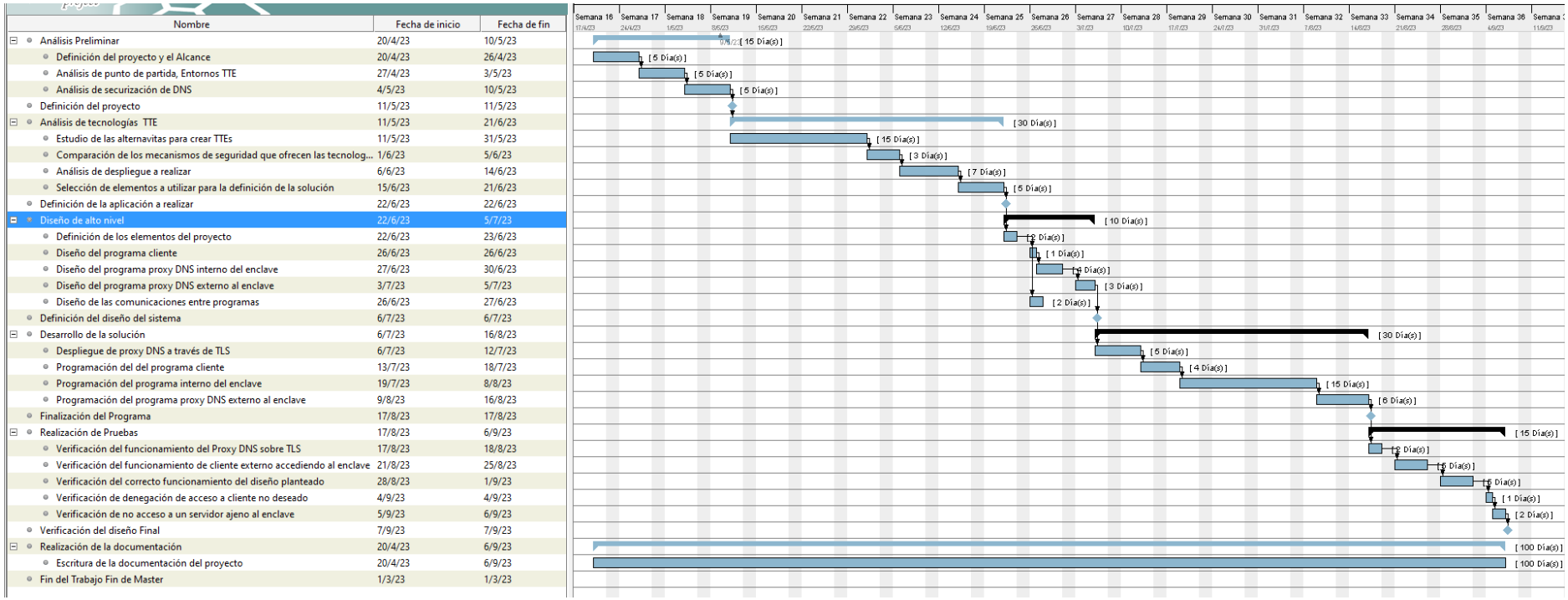
- **Escritura de la documentación del proyecto** (120h).

11.2.1 Resumen de fases del proyecto

A continuación, se muestra una tabla con las tareas indicadas y la duración que tienen en días utilizadas para la realización de la planificación.

Código	Nombre de tarea	Duración
P.T.1	Análisis Preliminar	15 días
T.1.1	Definición del proyecto y el Alcance	5 días
T.1.2	Análisis de punto de partida, Entornos TTE	5 días
T.1.3	Análisis de securización de DNS	5 días
H1	**Definición del proyecto**	
P.T.2	Análisis de tecnologías TTE	30 días
T.2.1	Estudio de las alternativas para crear TTEs	15 días
T.2.2	Comparación de los mecanismos de seguridad que ofrecen las tecnologías	3 días
T.2.3	Análisis de despliegue a realizar	7 días
T.2.4	Selección de elementos a utilizar para definición de la solución	5 días
H2	**Definición de la aplicación a realizar**	
P.T.3	Diseño de alto nivel	10 días
T.3.1	Definición de los elementos del proyecto	2 días
T.3.2	Diseño del programa cliente	1 día
T.3.3	Diseño del programa proxy DNS interno del enclave	4 días
T.3.4	Diseño del programa proxy DNS externo al enclave	3 días
T.3.5	Diseño de las comunicaciones entre programas	2 días
H3	**Definición del diseño del sistema**	
P.T.4	Desarrollo de la solución	30 días
T.4.1	Despliegue de proxy DNS a través de TLS	5 días
T.4.2	Programación del del cliente	4 días
T.4.3	Programación del programa interno del enclave	15 días
T.4.4	Programación del proxy DNS externo al enclave	6 días
H4	**Finalización del Programa**	
P.T.5	Realización de Pruebas	15 días
T.5.1	Verificación del funcionamiento del Proxy DNS sobre TLS	2 días
T.5.2	Verificación del funcionamiento de cliente externo accediendo al enclave	5 días
T.5.3	Verificación del correcto funcionamiento del diseño planteado	5 días
T.5.4	Verificación de denegación de acceso a cliente no deseado	1 día
T.5.5	Verificación de no acceso a un servidor ajeno al enclave	2 días
H5	**Verificación del diseño Final**	
P.T.6	Realización de la documentación	100 días
T.6.1	Escritura de la documentación del proyecto	100 días

Tabla 11: Resumen de las fases del proyecto



Nombre	Fecha de inicio	Fecha de fin
Análisis Preliminar	20/4/23	10/5/23
Definición del proyecto y el Alcance	20/4/23	26/4/23
Análisis de punto de partida, Entornos TTE	27/4/23	3/5/23
Análisis de securización de DNS	4/5/23	10/5/23
Definición del proyecto	11/5/23	11/5/23
Análisis de tecnologías TTE	11/5/23	21/6/23
Estudio de las alternavitas para crear TTEs	11/5/23	31/5/23
Comparación de los mecanismos de seguridad que ofrecen las tecnolog...	1/6/23	5/6/23
Análisis de despliegue a realizar	6/6/23	14/6/23
Selección de elementos a utilizar para la definición de la solución	15/6/23	21/6/23
Definición de la aplicación a realizar	22/6/23	22/6/23
Diseño de alto nivel	22/6/23	5/7/23
Definición de los elementos del proyecto	22/6/23	23/6/23
Diseño del programa cliente	26/6/23	26/6/23
Diseño del programa proxy DNS interno del enclave	27/6/23	30/6/23
Diseño del programa proxy DNS externo al enclave	3/7/23	5/7/23
Diseño de las comunicaciones entre programas	26/6/23	27/6/23
Definición del diseño del sistema	6/7/23	6/7/23
Desarrollo de la solución	6/7/23	16/8/23
Despliegue de proxy DNS a través de TLS	6/7/23	12/7/23
Programación del del programa cliente	13/7/23	18/7/23
Programación del programa interno del enclave	19/7/23	8/8/23
Programación del programa proxy DNS externo al enclave	9/8/23	16/8/23
Finalización del Programa	17/8/23	17/8/23
Realización de Pruebas	17/8/23	6/9/23
Verificación del funcionamiento del Proxy DNS sobre TLS	17/8/23	18/8/23
Verificación del funcionamiento de cliente externo accediendo al enclave	21/8/23	25/8/23
Verificación del correcto funcionamiento del diseño planteado	28/8/23	1/9/23
Verificación de denegación de acceso a cliente no deseado	4/9/23	4/9/23
Verificación de no acceso a un servidor ajeno al enclave	5/9/23	6/9/23
Verificación del diseño Final	7/9/23	7/9/23
Realización de la documentación	20/4/23	6/9/23
Escritura de la documentación del proyecto	20/4/23	6/9/23
Fin del Trabajo Fin de Master	1/3/23	1/3/23

Figura 47: Diagrama de Gantt de la planificación del proyecto

12 PRESUPUESTO

En el presente apartado, se describe el coste de la realización del presente Trabajo de Fin de Máster. El objetivo de este punto es cuantificar en términos económicos la inversión necesaria a realizar para desarrollar el trabajo y determinar si es viable a nivel económico o no.

El presupuesto que se indica a continuación está formado por cuatro partidas que se separan en de la siguiente manera.

12.1 Recursos humanos

En esta partida tiene en cuenta las horas internas necesarias para la realización del proyecto completo. Se tendrán en cuenta el número de horas necesarias para la realización del proyecto de cada uno de los dos trabajadores que completan el equipo. En la siguiente tabla se muestra el desglose de los sueldos horarios y las horas dedicadas de cada uno de los participantes del equipo:

Concepto	Horas (h)	Coste/horas (€/h)	Coste total (€)
Ingeniero Junio	600	14	8.400,00 €
Ingeniero Senior	30	60	1.800,00 €
SUBTOTAL			**10.200,00 €**

Tabla 12: Recursos humanos

12.2 Amortizaciones

En este apartado se indica el coste necesario a amortizar para poder realizar el presente proyecto. En este subapartado, se tendrán en cuenta las licencias necesarias para llevar a cabo el proyecto y el equipamiento que ha sido necesario utilizar.

Concepto	Coste (€)	Vida útil (h)	Horas usadas (h)	coste/horas (€/h)	Coste total (€)
Licencia Microsoft Office	69	8760	150	0,01	1,18 €
Ordenador Junior	700	26280	600	0,03	15,98 €
Ordenador Senior	1500	26280	30	0,06	1,71 €
Licencia Windows	19,9	26280	300	0,001	0,23 €
SUBTOTAL					**19,10 €**

Tabla 13: Amortizaciones

12.3 Gastos

Por último, la última partida que tenemos definida es la partida de gastos. Dado que la realización del proyecto se ha acabado realizando desde un ordenador portátil personal y una máquina virtual de Azure alquilada, los únicos gastos que se han detectado en este proyecto es el material de oficina.

Concepto	Coste total (€)
Material oficina	30,00 €
Máquina Virtual Azure	64,11 €
Almacenamiento	3,96 €
Programación de la máquina	5,89 €
Conexión	5,30 €
SUBTOTAL	**109,26 €**

Tabla 14: Gastos

12.4 Presupuesto final

Por último, para definir el presupuesto final necesario para poder financiar el presente proyecto hay que realizar la suma del coste total de cada una de las partidas definidas en este apartado. Por último, se añade un coste del 10% del presupuesto total como imprevistos con la intención de manejar un margen de error a la hora de realizar los cálculos o por si hay alguna otra partida que no se haya tenido en cuenta. La siguiente tabla muestra el presupuesto general del proyecto:

Concepto	Coste total (€)
Horas internas	10.200,00 €
Amortizaciones	19,10 €
Gastos	109,26 €
imprevistos (10%)	1.031,84 €
SUBTOTAL	**11.361,19 €**

Tabla 15: Presupuesto del proyecto

13 CONCLUSIONES

Tras la realización del proyecto, se puede concluir que se han cumplido con los objetivos propuestos al inicio del trabajo. Por un lado, se ha realizado un análisis exhaustivo para de las diferentes tecnologías relacionadas con los Entornos de Ejecución Seguros y poder relacionarlos para dotar de mecanismos de seguridad, como la confidencialidad y la integridad al protocolo DNS. Además, las pruebas realizadas para el proyecto muestran que, a parte de estos mecanismos ya comentados, se ofrecen otros mecanismos de seguridad como autenticación y autorización de los actores.

Además, el obligar al programa cliente a realizar la petición al proxy DNS a través del enclave seguro genera que el cliente tenga que realizar dos procesos de acceso al proxy, lo que permite reducir de manera notoria la vulnerabilidad de hacer que el usuario se conecte a un servidor malicioso.

El comunicar el usuario con el enclave a través de un canal socket sobre TLS ofrece que el cliente reciba la dirección IP del nombre de dominio deseado sin necesidad de crear una comunicación DNS. Esto dificulta en gran medida ser víctima de un ataque de “Man in the Middle” o usurpación de la identidad, ya que se evita la conexión directa con el proxy y mediante un protocolo ajeno al DNS, por lo que no necesitaría de salida DNS para traducir el nombre de dominio y siempre lo realizaría a través de un programa ejecutado en un entorno fiable.

Por otro lado, la realización del estudio de seguridad ha permitido identificar de forma más clara las vulnerabilidades a proteger del protocolo DNS. Lo que permite enfocar el desarrollo a realizar con el objetivo claro de ofrecer seguridad con la intención de eliminar dichas vulnerabilidades del protocolo o, reducir el impacto que puedan generar sobre el protocolo.

Respecto al desarrollo, se llega a la conclusión de las infinitas posibilidades de servicios que ofrecen los portales como Microsoft Azure, ya que el alquiler de una de las maquinas que tienen disponibles a un coste razonable. Para el despliegue del desarrollo se ha realizado el alquiler de una de estas máquinas virtuales y, el coste que tienen se amortiza en el ahorro de horas a dedicar a la solución de los problemas de compatibilidad que pueden surgir en las librerías SGX en la realización del proyecto.

De cara al futuro, este trabajo ha permitido obtener conocimientos técnicos acerca de nuevos modelos de protección como la protección a nivel de hardware que ofrecen los enclaves. Estos mecanismos ofrecen un nivel de seguridad más estricto a las aplicaciones que se ejecutan dentro del enclave. Además, se ha aprendido que estos entornos se pueden utilizar para proteger cualquier tipo de protocolo o programa que esté manejando información sensible.

Por último, otra de las conclusiones que se sacan con el presente trabajo es la cantidad de vulnerabilidades tiene actualmente protocolo DNS convencional. Además, al ser un protocolo transparente al usuario, los usuarios no son conscientes del riego que puede generar que un tercero pueda manejar esta información. Por eso, es importante proteger estos protocolos, aunque no se sepa que se están ejecutando en un segundo plano, ya que se puede estar dando información sensible a terceras personas sin saberlo.

14 BIBLIOGRAFÍA

[1] M. d. Interior, Balance de criminalidad 4 trimestre, 2022.

[2] M. d. Interior, Balance de criminalidad 1 trimestre, 2023.

[3] ICANN, «ICAN,» [En línea]. Available: https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en.

[4] «RFC 4033 DNS Security Introduction and Requirements,» Marzo 2005.

[5] «RFC 3833: Threat Analysis of the Domain Name System (DNS),» Agosto 2004.

[6] «Enclave Aware Containers with Intel SGX,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/enclave-aware-containers.

[7] Z. Y. T. D. a. H. X. SHUFAN FEI, «Security Vulnerabilities of SGX and Countermeasures: A survey,» *ACM Computing Surveys,* Vols. %1 de %2Vol. 54,, nº No. 6, Article 126., Julio 2021.

[8] Microsoft, «Application enclave development,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/application-development.

[9] «https://learn.microsoft.com/en-us/azure/confidential-computing/enclave-aware-containers,» [En línea].

[10] F. Z. S. L. y. W. S. Saeid Mofrad, «A Comparison Study of Intel SGX and AMD Memory Encryption Technology,» *HASP '18,* Junio 2018.

[11] J. P. T. W. David Kaplan, «AMD MEMORY ENCRYPTION,» Octubre 2018.

[12] «AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More,» Enero 2020.

[13] «Learn the architecture - TrustZone for AArch64,» 2020-2021.

[14] «Learn the architecture - Realm Management Extension,» 2021.

[15] «Learn the architecture - AArch64 Exception Model,» 2022.

[16] «Learn the architecture - Introducing Arm Confidential Compute Architecture,» 2022-223.

[17] J. Ménétrey, C. Göttel, M. Pasin, P. Felber y V. Schiavoni, «An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments,» *SysTEX '22 Workshop,* Abril 2022.

[18] «QEMU documentation,» [En línea]. Available: https://www.qemu.org/documentation/.

[19] «QUEMU: Software Guard eXtensions (SGX),» [En línea]. Available: https://www.qemu.org/docs/master/system/i386/sgx.html.

[20] «Precios de Azure Dedicated Host,» [En línea]. Available: https://azure.microsoft.com/es-es/pricing/details/virtual-machines/dedicated-host/.

[21] «Github: intel,» [En línea]. Available: https://github.com/intel/linux-sgx/tree/master.

[22] «Open Enclave SDK,» [En línea]. Available: https://openenclave.io/sdk/.

[23] «Github: edgelesssys/ego,» [En línea]. Available: https://github.com/edgelesssys/ego.

[24] «Github: intel/quemu-sgx,» [En línea]. Available: https://github.com/intel/qemu-sgx.

[25] N. Buchner, H. Kinkelin y F. Rezabe, «Survey on Trusted Execution Environments,» *Seminar IITM WS 21/22,* Mayo 2022.

[26] «ISO 31000,» 2018.

[27] «Install and configure xrdp to use Remote Desktop with Ubuntu,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/linux/use-remote-desktop?tabs=azure-cli.

[28] «ego,» [En línea]. Available: https://pkg.go.dev/github.com/edgelesssys/ego#section-readme.

[29] «Github: miekg/dns,» [En línea]. Available: https://github.com/miekg/dns.

ANEXO 1: CÓDIGO DEL PROGRAMA

Programa Cliente:

```
package main

import (
	"bytes"
	"crypto/tls"
	"encoding/binary"
	"encoding/hex"
	"errors"
	"flag"
	"fmt"
	//"io/ioutil"
	//"net"

	"github.com/edgelesssys/ego/attestation"
	"github.com/edgelesssys/ego/eclient"
	"github.com/miekg/dns"
)

var signer []byte

func main() {
	signerArg := flag.String("s", "", "signer ID")
	serverAddr := flag.String("a", "localhost:3085", "server address")
	flag.Parse()

	// get signer command line argument
	var err error
	signer, err = hex.DecodeString(*signerArg)
	if err != nil {
		panic(err)
	}
	if len(signer) == 0 {
		flag.Usage()
		return
	}

	// Create a TLS config that verifies a certificate with embedded report.
	tlsConfig := eclient.CreateAttestationClientTLSConfig(verifyReport)

	// Connect to the secure shocket.
	conn, err := tls.Dial("tcp", *serverAddr, tlsConfig)
	if err != nil {
		panic(err)
	}
	defer conn.Close()

	fmt.Println("Conexión con el servidor establecida.")
```

```
	fmt.Print("Introduzca el nombre de dominio a traducir: ")
	var input string
	fmt.Scanln(&input)

	_, err = conn.Write([]byte(input))
	if err != nil {
		panic(err)
	}
	response := make([]byte, 1024)
	n, err := conn.Read(response)
	if err != nil {
		panic(err)
	}

	var responseMsg dns.Msg
	err = responseMsg.Unpack(response[:n])
	if err != nil {
		fmt.Println("Error al deserializar la respuesta:", err)
		return
	}

	fmt.Println("Respuesta DNS:")
	fmt.Println(responseMsg.String()) // Esto mostrará la respuesta como una consulta DNS legible

}

func verifyReport(report attestation.Report) error {
	// You can either verify the UniqueID or the tuple (SignerID, ProductID, SecurityVersion, Debug).

	if report.SecurityVersion < 2 {
		return errors.New("invalid security version")
	}
	if binary.LittleEndian.Uint16(report.ProductID) != 1234 {
		return errors.New("invalid product")
	}
	if !bytes.Equal(report.SignerID, signer) {
		return errors.New("invalid signer")
	}

	// For production, you must also verify that report.Debug == false
	return nil
}
```

Programa Enclave

```
package main

import (
	"fmt"
	"net"
	"crypto/tls"

	"github.com/edgelesssys/ego/enclave"
)

var conn1, conn2 net.Conn // Mantén referencias a los dos sockets

func main() {
	// Create a TLS config with a self-signed certificate and an embedded report.
	tlsCfg1, err := enclave.CreateAttestationServerTLSConfig()
	if err != nil {
		panic(err)
	}

	tlsCfg2, err := enclave.CreateAttestationServerTLSConfig() // Crear una segunda configuración TLS
	if err != nil {
		panic(err)
	}

	listener1, err := tls.Listen("tcp", "0.0.0.0:3080", tlsCfg1)
	if err != nil {
		panic(err)
	}
	defer listener1.Close()

	fmt.Println("socket 1 escuchando...")

	listener2, err := tls.Listen("tcp", "0.0.0.0:3081", tlsCfg2)
	if err != nil {
		panic(err)
	}
	defer listener2.Close()

	fmt.Println("socket 2 escuchando...")

	conn1, err = listener1.Accept()
	if err != nil {
		panic(err)
	}
```

```
	conn2, err = listener2.Accept()
	if err != nil {
		panic(err)
	}

	go handleConnection(conn1, conn2)
	go handleConnection(conn2, conn1)

	select {}
}

func handleConnection(srcConn, destConn net.Conn) {
	defer srcConn.Close()

	buffer := make([]byte, 1024)
	for {
		n, err := srcConn.Read(buffer)
		if err != nil {
			fmt.Println(err)
			return
		}

		receivedData := buffer[:n]
		fmt.Printf("Menaje recibido desde %s\n", srcConn.RemoteAddr())

		_, err = destConn.Write(receivedData)
		if err != nil {
			fmt.Println(err)
			return
		}
	}
}
```

Programa Proxy

```
package main

import (
	"bytes"
	"crypto/tls"
	"encoding/binary"
	"encoding/hex"
	"errors"
	"flag"
```

```
	"fmt"
	"strings"
	"strconv"
	//"io/ioutil"
	//"net"

	"github.com/edgelesssys/ego/attestation"
	"github.com/edgelesssys/ego/eclient"
	"github.com/miekg/dns"
)

var signer []byte

func main() {
	signerArg := flag.String("s", "", "signer ID")
	serverAddr := flag.String("a", "localhost:3081", "server address")
	flag.Parse()

	// get signer command line argument
	var err error
	signer, err = hex.DecodeString(*signerArg)
	if err != nil {
		panic(err)
	}
	if len(signer) == 0 {
		flag.Usage()
		return
	}

	// Create a TLS config that verifies a certificate with embedded report.
	tlsConfig := eclient.CreateAttestationClientTLSConfig(verifyReport)

	// Connect to the secure shocket.
	conn, err := tls.Dial("tcp", *serverAddr, tlsConfig)
	if err != nil {
		panic(err)
	}
	defer conn.Close()

	fmt.Println("Conexión establecida con el servidor.")

	response := make([]byte, 512)
	n, err := conn.Read(response)
	if err != nil {
		panic(err)
	}
	fmt.Printf("Recibido del enclave: %s\n", response[:n])

	//fmt.Print("Server response bytes: ")
```

```
	cleanedBytes := removeNullBytes(response)

	// Convertir los bytes en strings
	byteStrings := bytesToStrings(cleanedBytes)


	var asciiResult strings.Builder
	for _, hexStr := range byteStrings {
		asciiResult.WriteString(hexToASCII(hexStr))
	}

	resultString := asciiResult.String()



	tlsConfig2 := &tls.Config{
		ServerName: "dns.google",
	}
	dnsClient := &dns.Client{
		Net:      "tcp-tls",
		TLSConfig: tlsConfig2,
	}


	query := new(dns.Msg)
	query.SetQuestion(resultString, dns.TypeA)

	resp, _, err := dnsClient.Exchange(query, "8.8.8.8:853")
	if err != nil {
		fmt.Println("Error al enviar la consulta DNS:", err)
		return
	}



	responseBytes, err := resp.Pack()
	if err != nil {
		fmt.Println("Error al serializar la respuesta:", err)
		return
	}

	_, err = conn.Write(responseBytes)
	if err != nil {
		panic(err)
	}
}


func verifyReport(report attestation.Report) error {
```

```
	// You can either verify the UniqueID or the tuple (SignerID, ProductID, SecurityVersion, Debug).

	if report.SecurityVersion < 2 {
		return errors.New("invalid security version")
	}
	if binary.LittleEndian.Uint16(report.ProductID) != 1234 {
		return errors.New("invalid product")
	}
	if !bytes.Equal(report.SignerID, signer) {
		return errors.New("invalid signer")
	}
	return nil
}

func bytesToStrings(bytes []byte) []string {
	strings := make([]string, len(bytes))
	for i, b := range bytes {
		strings[i] = fmt.Sprintf("%02x", b)
	}
	return strings
}

func removeNullBytes(input []byte) []byte {
	var output []byte
	for _, b := range input {
		if b != 0x00 {
			output = append(output, b)
		}
	}
	return output
}

func hexToASCII(hexStr string) string {
	value, _ := strconv.ParseUint(hexStr, 16, 8)
	return string(value)
}
```