UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Análisis de seguridad del protocolo DLMS/COSEM en el contexto de Smart Grids

Joaquín Márquez

Gabriel Rodríguez

Proyecto de grado en Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay

Febrero de 2020

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Análisis de seguridad del protocolo DLMS/COSEM en el contexto de Smart Grids

Joaquín Márquez

Gabriel Rodríguez

Proyecto de grado presentado como parte de los requisitos necesarios para la obtención del título de Ingeniero en Computación de Facultad de Ingeniería de la Universidad de la República

Directores:

Gustavo Betarte

Eduardo Grampín

Codirector:

Juan Diego Campo

Montevideo – Uruguay

Febrero de 2020

INTEGRANTES DEL TRIBUNAL DE DEFENSA

Patricia Hernández

Leonardo Vidal

Felipe Zipitría

Montevideo – Uruguay

Febrero de 2020

RESUMEN

En los últimos tiempos el surgimiento de soluciones de “IoT” o “Internet de las cosas” se ha incrementado notablemente. IoT agrega inteligencia a la vida y al trabajo de las personas, ofrece a las empresas y negocios nuevos beneficios y oportunidades, automatizando procesos y reduciendo costos. Muchas empresas de energía eléctrica a lo largo del mundo han comenzado a implementar soluciones de Smart Grid. Esto conlleva el surgimiento de nuevas problemáticas de seguridad informática, entre las que se incluyen garantizar la privacidad de los datos personales de los consumidores y la integridad y confidencialidad de los datos de consumo de energía, e incluso la posibilidad de ataques que atenten contra la disponibilidad del servicio.

Una de estas empresas es UTE, la cual se encuentra en proceso de despliegue de una solución IoT para un sistema de recolección de medidas de consumo eléctrico. El proyecto se enmarca en una colaboración entre UTE e investigadores de la Facultad de Ingeniería de la UdelaR, que tuvo como objetivo analizar la seguridad de dicho sistema. El proyecto se centró en un análisis de seguridad del protocolo DLMS/-COSEM, el cual es ampliamente adoptado en el mundo para la comunicación con medidores inteligentes, y en particular, el utilizado en la comunicación dentro de la solución de Smart Grid de UTE.

En este informe se presenta el estado del arte del área, y se analiza y describe la implementación de un prototipo para una herramienta que permita la evaluación de las características de seguridad del sistemas de recolección de medidas de consumo de energía en forma automatizada, en particular mediante el análisis de vulnerabilidades del protocolo DLMS/COSEM.

Palabras claves:
Smart Grid, DLMS/COSEM, Smart Meter, AMI, IoT.

Tabla de contenidos

Capítulo 1

Introducción

En las últimas décadas hemos sido testigos del surgimiento de un nuevo paradigma tecnológico al que denominamos “Internet de las cosas” o “IoT”. El número de dispositivos IoT en el mundo crece a un ritmo mayor año a año, haciendo a las soluciones IoT cada vez más penetrantes en la vida diaria, tanto en áreas de menor importancia como en infraestructuras críticas.

Un sistema IoT puede visualizarse como una red global de objetos conectados entre sí y a Internet que tienen la capacidad de interactuar e intercambiar datos sin la necesidad de intervención humana. IoT no se trata solo de agrupar y conectar tecnologías y hardware, sino que como paradigma redefine la manera en la que visualizamos prácticamente cada aspecto de nuestra experiencia diaria, de los negocios y de la sociedad, permitiéndonos ver el potencial completo de la tecnología y empujar los límites de la creatividad y la innovación. IoT tiene como objetivo crear sistemas que generen más inteligencia y valor que la suma de sus partes.

La incidencia de IoT en áreas industriales, de servicios, entretenimiento, salud y otras no deja de expandirse. El avance tecnológico hace cada vez más posible el uso de IoT en soluciones a problemas que antes eran impensables. Surgen así nuevos mercados con nuevos negocios, donde los expertos estiman que gran parte de éstos utilizarán IoT de alguna manera en los productos y/o soluciones involucradas.

Si bien puede verse a IoT como un cambio tecnológico prometedor, para consolidarse no está ausente de desafíos por enfrentar, siendo la seguridad de los sistemas uno de los más críticos y complejos. Los avances tecnológicos que IoT aporta a los sistemas y soluciones también los expone en mayor medida a ataques y amenazas que pueden poner en riesgo su confidencialidad, integridad y/o disponibilidad. Los

consumidores depositan cada vez más confianza en los dispositivos, lo que aumenta la gravedad de los posibles efectos adversos y abre nuevas interrogantes sobre responsabilidades e impactos sociales.

Por lo general, garantizar seguridad en los sistemas IoT requiere esfuerzos adicionales, ya sea en recursos físicos o en dedicación. Muchos de estos sistemas, además, no fueron pensados desde un principio para ser conectados entre sí y a internet. Es así que al agregar funcionalidades de comunicación a dispositivos ya existentes que no las poseían se deberá ser sumamente cuidadoso.

Uno de los campos de aplicación de las soluciones IoT es en el de generación y distribución de energía eléctrica. Es allí donde surge el concepto de "Smart Grid". Las redes eléctricas se encuentran en un proceso de evolución, donde a la infraestructura ya existente se le agregan capacidades y funcionalidades en busca de mejorar los niveles de eficiencia y calidad del servicio, utilizando para ello las crecientes capacidades de procesamiento e inteligencia de la tecnología actual.

El despliegue de Smart Grids se ha convertido en tema de gran interés en el mundo. Algunos países invierten fuertemente en investigación relacionada a estos temas debido a todos los beneficios que ofrece a las compañías y consumidores. La funcionalidad principal de este tipo de sistemas es la provisión de información casi en tiempo real sobre el consumo de energía. De esta manera, el sistema ayuda a balancear la generación y distribución de energía en función de la demanda, y también permite que los usuarios adapten dinámicamente su consumo.

Sin embargo, pensar en un futuro donde el uso de Smart grids se haya extendido genera preocupaciones relacionadas a la seguridad y privacidad. Entre éstas se encuentra la posibilidad de divulgación de información personal de los consumidores, de engaños al sistema, de reportes de datos de consumo falso e incluso de ataques a la seguridad nacional al intentar dar de baja partes de la red eléctrica.

Este proyecto de grado se enmarca dentro de una colaboración entre UTE e investigadores de la Facultad de Ingeniería de la UdelaR. Motivado por el interés de UTE en desplegar una solución de Smart Grid que permita controlar y administrar dispositivos instalados en sus clientes, se realizó una actividad de investigación cuyo objetivo fue identificar y desarrollar procedimientos metodológicos e instrumentos técnicos que le permitan incorporar garantías de la corrección de las soluciones, en particular en relación a las propiedades de seguridad que deberían ser garantizadas por éstas.

En este contexto, se trabajó sobre el sistema de recolección de datos, el cual

recolecta los datos de la red de medidores inteligentes y los transmite utilizando la plataforma de IoT de ANTEL. Nuestro trabajo se centró en el estudio de las características de seguridad de los protocolos de comunicación en la red de medidores y concentradores, en particular del protocolo DLMS/COSEM por ser el utilizado por los medidores en la solución de UTE y por tratarse de un estándar en las soluciones de este tipo en el resto del mundo.

En el comienzo del proyecto se plantearon los siguientes objetivos generales:

- Relevar el tema mediante el estudio de la bibliografía disponible.
- Contribuir en el avance del conocimiento en el área.
- Colaborar en el cumplimiento de los objetivos del proyecto con UTE.

Luego de una etapa de análisis inicial, se decidieron los siguientes objetivos específicos:

- Estudiar el protocolo DLMS/COSEM.
- Relevar las problemáticas de seguridad del protocolo.
- Contribuir a la detección de vulnerabilidades del protocolo que puedan ser utilizadas en la realización de ataques.
- Desarrollar una herramienta que brinde soporte automatizado al proceso de detección de vulnerabilidades en el sistema de medidas.

En el presente informe se presentan los resultados obtenidos durante la realización del proyecto. El capítulo 2 contiene un resumen del estado del arte recabado sobre IoT, Smart Grids y el protocolo DLMS/COSEM, centrándose en sus aspectos de seguridad. El capítulo 3 desarrolla sobre el análisis de la herramienta que se tuvo como objetivo implementar. El capítulo 4 presenta la etapa de diseño e implementación y el capítulo 5 su proceso de verificación. En el capítulo 6 se detallan brevemente las distintas etapas del proyecto. En el capítulo 7 se presentan las conclusiones y el trabajo a futuro.

Por último, adjunto al presente informe, se presenta un informe adicional en el cual se detalla el trabajo realizado en el contexto del proyecto conjunto con UTE. El contenido de dicho informe se encuentra abarcado por un acuerdo de confidencialidad, y es por ello que no se presenta en conjunto con el resto de los resultados del proyecto.

Capítulo 2

Estado del arte

En la presente sección se presenta un resumen del estado del arte generado a partir de la bibliografía consultada. La versión completa se presenta en el documento adjunto.

2.1. IoT

2.1.1. Definiciones

La expresión "Internet de las cosas" está ganando cada vez más y más popularidad. Fue utilizada por primera vez en 1999 por el británico Kevin Ashton, pionero en tecnología e investigador en el MIT, y se encuentra entre las tecnologías estratégicas con mayor tendencia a nivel mundial, abarcando cada vez más dispositivos e incrementando sus implicancias económicas año a año. [1]

A la hora de definir IoT, varias instituciones y agentes han realizado aportes y propuesto definiciones propias pero aún no se dispone de una única definición acordada y consensuada por todos éstos. Entre las definiciones propuestas se encuentran las que se presentan a continuación.

Por un lado, la agencia ENISA (Agencia en Ciberseguridad de la Unión Europea) define IoT como:

"Ecosistema ciberfísico de sensores y agentes interconectados, que da lugar a la toma inteligente de decisiones". [2]

El comité ISO/IEC, por su lado, brinda la siguiente definición:

"Infraestructura donde objetos, personas, sistemas y fuentes de información se encuentran interconectados entre sí y con servicios inteligentes que les permiten procesar información del mundo físico y virtual y reaccionar". [3]

El instituto IEEE también publica su propia definición de IoT en [4], describiéndolo como:

"Red cableada o inalámbrica de dispositivos conectados, únicos e identificables que son capaces de procesar datos y comunicarse entre ellos, con y sin el involucramiento humano".

Es interesante observar cómo, si bien las definiciones surgen desde enfoques distintos, el concepto de conectividad entre objetos se encuentra presente en todas ellas. Los objetos en IoT no necesitan una conexión a internet propiamente dicha. El uso de la palabra Internet en IoT debe ser vista solamente como una generalización que busca transmitir la idea de conectividad y no como un requerimiento técnico necesario. [2]

2.1.2. Consideraciones iniciales

Según el último reporte sobre tecnologías móviles de la compañía Ericsson de junio de 2019, en 2018 se contabilizaron alrededor de 8.600 millones de dispositivos IoT en todo el mundo, y se estima que dicha cifra aumente a 22.300 millones para el año 2024. [5]

Según afirman los autores en [1], nos encontramos muy cerca de ser testigos del surgimiento de un mega mercado, donde mercados de distinta índole van a converger de forma constante. Actividades diarias que en un principio no tenían puntos de contacto, se ven interconectadas exponenciando sus capacidades, creando nuevos formatos y modelos de negocio.

Uno de los tantos desafíos de IoT es la necesidad de entender y mejorar la interacción humana con las máquinas. IoT generará nuevas dificultades que los autores de [1] llaman de "integridad contextual", principio que refiere a que no se espera, por parte del usuario, que información suministrada para ser utilizada en un contexto sea utilizada en otro. Existirá entonces algo así como un "contrato social" entre personas y objetos, donde todas sus ramificaciones éticas deberán ser consideradas.

Por otro lado, ENISA advierte en [2] que IoT conlleva desafíos adicionales. Estos nuevos ecosistemas aún se encuentran en una fase inmadura, enfrentándose a la fragmentación en la definición de estándares y a problemáticas de seguridad en un

entorno para nada homogéneo, donde cada mercado, industria y aplicación tiene un uso diferente y particular de la tecnología.

2.1.3. Componentes

Los ecosistemas IoT tienen un alto grado de complejidad, con un variado número de elementos que los componen con distintas características. En [2] los autores detallan y describen los mismos de la siguiente manera:

- *"Things" o "cosas"*
  En el contexto de IoT, una "cosa" es un objeto físico o virtual capaz de ser identificado e integrado en una red de comunicación. Una de sus características principales es su capacidad de comunicación con otras "cosas", es necesario que los objetos puedan intercambiar datos a través de algún medio.

- *Toma inteligente de decisiones*
  Se incorpora la noción de que las cosas realizan acciones dentro del ecosistema. Es en este sentido en el que se requiere que exista una toma inteligente de decisiones que resulten en acciones específicas que brinden valor.

- *Sensores y "actuators"*
  En el nivel físico, los sensores pueden medir distintos indicadores físicos, químicos o biológicos definidos, y en el nivel digital, recolectan información sobre las redes y aplicaciones con las que interacciona o es parte. Por otro lado, los "actuators" son aquellos responsables de mover o controlar un sistema o mecanismo.

- *Sistemas embebidos*
  Los sensores y "actuators", además de poder ser parte del ecosistema por sí solos, pueden también encontrarse embebidos en sistemas con capacidades adicionales.

- *Comunicaciones*
  Dada la gran variedad de dispositivos y objetos que forman parte de este ecosistema, los sistemas de comunicación dependen de la capacidad tanto para transmitir como para recibir unidades de información de manera estructurada.

2.1.4. Requerimientos y desafíos

Al adentrarnos en la complejidad del paradigma IoT, nos encontramos ante una gran variedad de requerimientos, que a su vez pueden considerarse desafíos en sí mismos dada la realidad de constantes cambios en la cual opera IoT.

A continuación, se presentan los principales requerimientos o desafíos en los que IoT deberá continuar avanzando:

- *Interconectividad*
  Uno de los aspectos básicos de IoT es la necesidad de conectividad entre todos sus elementos. Las arquitecturas actuales de comunicación en IoT son muy variadas, utilizando distintas combinaciones de opciones en cuanto a redes, dispositivos y servicios. Actualmente, gran parte de las comunicaciones se realizan a través de VPNs o redes públicas dedicadas. [6]

- *Heterogeneidad*
  En IoT, nos referimos con heterogeneidad a la diversidad de dispositivos y su performance, redes, protocolos, plataformas, políticas, dominios de aplicación, entre otros, por la cual está compuesta [7].

- *Interoperabilidad*
  Se define como la habilidad para hacer cooperar distintos sistemas y dispositivos de una manera eficiente [8]. La interoperabilidad necesita ser asegurada entre los sistemas. [9]

- *Escalabilidad*
  La escalabilidad es la habilidad de un sistema o red de manejar la escala de crecimiento de cualquier ambiente sin consecuencias en su rendimiento. [8]

- *Procesamiento de datos*
  El procesamiento inteligente de los datos obtenidos es una funcionalidad clave de un sistema IoT. La habilidad de distribuir sensores ampliamente y recopilar datos de manera rápida y efectiva facilita nuevas formas de colaboración. [6]

- *Calidad del servicio (QoS)*
  La calidad del servicio, como un requerimiento no funcional, es la capacidad de proveer un servicio satisfactorio por los distintos proveedores y sistemas. [10]

- *Seguridad*
  Las amenazas y riesgos relacionados a los dispositivos, sistemas y servicios IoT

son variados, y evolucionan rápidamente. En la siguiente sección se profundizará en este punto, introduciendo más en detalle la problemática de seguridad en el contexto de sistemas IoT.

2.2. Seguridad en IoT

Los dispositivos son el blanco de numerosos ataques que tienen el potencial de poner en peligro la privacidad de las personas y amenazar la seguridad pública. Es así que la seguridad informática (security) es una de las mayores preocupaciones en relación a IoT, que necesita ser abordada conjunto con la necesidad de seguridad física (safety), dado que ambas están intrínsecamente relacionadas con el mundo físico. [2]

Cuando hablamos de seguridad nos referimos a la condición de un sistema de estar protegido contra el acceso no intencionado o no autorizado, modificaciones y daños. El comportamiento seguro de un sistema no es total. Ningún sistema IoT puede comportarse de forma segura en todos los contextos. [11]

Las propiedades que necesitan ser garantizadas para proveer la seguridad de la información y los recursos del sistema son confidencialidad, integridad y disponibilidad, también referidos con el acrónimo CIA (confidentiality, integrity and availability). La propiedad de confidencialidad refiere a que la información no sea accesible ni revelada a individuos, entidades o procesos no autorizados. La propiedad de integridad se asegura de que ninguna modificación o destrucción de información se lleve a cabo. Por último, la disponibilidad es la propiedad que refiere a brindar acceso a la información de forma oportuna, confiable y a demanda, a los usuarios autorizados. [11]

Es importante observar que las contramedidas tradicionales de seguridad no pueden ser directamente aplicadas a las tecnologías IoT debido a los diferentes estándares y capas de comunicación involucradas.[12]

2.2.1. Requerimientos de seguridad

Como ya se ha mencionado anteriormente, IoT se caracteriza por estar conformado por tecnologías heterogéneas que proveen servicios innovadores en distintos dominios de aplicación. En este escenario, la satisfacción de requerimientos de seguridad y privacidad juega un rol fundamental. [12]

Es así que el paradigma IoT presenta nuevos desafíos que hacen necesario que se definan requerimientos de seguridad distintos a los ya conocidos en el contexto IT. En [6] los autores analizan los distintos requerimientos de IoT, y presentan los siguientes requerimientos en el contexto de la seguridad de los sistemas. Pocos sistemas IoT actuales cumplen estos requerimientos de seguridad, pero es importante definirlos y promoverlos debido a la criticalidad de los desafíos de IoT hacia el futuro. Algunos de los requerimientos presentados son:

- Políticas de seguridad end-to-end y capacidad de gestión de riesgos
- Capacidad de monitoreo de los dispositivos y de detección de anomalías.
- Capacidad de gestión de los identificadores del sistema
- Seguridad adaptativa, receptiva y cooperativa. Capacidad de adaptación, aprendizaje e incorporación de nueva información sobre amenazas.
- Capacidad de responder rápida y apropiadamente a amenazas y ataques, mitigando el mayor daño posible.
- Capacidad de diagnosticar problemas cooperativamente e implementar planes de mitigación y prevención entre los distintos subsistemas del sistema.
- A nivel del borde del sistema, capacidad de autenticar y autorizar los distintos productos y dispositivos, y determinar su identidad, a su vez que la capacidad de controlar el acceso desde y hacia dichos dispositivos en base a la misma.
- Los dispositivos necesitarán capacidades cada vez mayores de resiliencia y tolerancia a fallas.

Los autores de [6] además presentan en detalle los siguientes requerimientos de seguridad, con énfasis en el IoT del futuro:

- *Identificadores seguros y gestión de los mismos*
  Se requiere de tecnologías escalables y eficientes para identificar dispositivos y objetos inteligentes. Nuevas tecnologías son necesarias para asignar identidades únicas a las cosas, éstas deberán ser escalables y no deberán permitir la clonación ni suplantación de identidad.

- *Preservar la privacidad en contextos multifacéticos y dinámicos*
  Los sistemas IoT son más que la suma de sus partes, por lo que los problemas en la privacidad se vuelven aún más complejos. Las consideraciones sobre la privacidad que pueden hacerse a bajo nivel pueden diferir de las preocupaciones generadas a nivel de la aplicación o del análisis de datos, y las violaciones de privacidad en cualquier nivel afectan al sistema entero.

- *Establecimiento de confianza*
  Las infraestructuras centrales de establecimiento de confianza resultan inaccesibles para la mayorías de los escenarios en IoT, donde la confianza debe ser establecida a demanda con pares que no fueron previamente registrados y que son desconocidos, sin la interacción de un usuario. Es así que algoritmos nuevos y más ligeros de establecimiento de confianza son requeridos.

- *Análisis de amenazas y gestión de riesgos*
  Los sistemas IoT necesitan ser capaces de adaptarse agregando contramedidas preventivas al sistema cada vez que una nueva amenaza es identificada, y además responder apropiadamente y mitigar el daño causado por éstas.

- *Gestión de seguridad continua*
  Componentes vulnerables del sistema IoT pueden violar requerimientos de seguridad de cualquiera de las partes involucradas. Conducir auditorías de seguridad en IoT requiere de un acercamiento diferente capaz de detectar continuamente cambios en los servicios IoT, evaluando su impacto en el nivel de seguridad en tiempo real.

2.2.2. Amenazas y riesgos

El número de amenazas de seguridad que afectan a los dispositivos IoT ha aumentado en los últimos años. La figura 2.1 ilustra algunos de los incidentes de seguridad, en el contexto de IoT, más importantes que han sido descubiertos y/o que tomaron lugar desde 2009, demostrando el gran crecimiento que tuvieron. [2]

Es así que el análisis de riesgos es vital en el contexto de IoT. El riesgo, efecto de la incertidumbre, toma en cuenta la probabilidad de que un evento ocurra conjunto con el impacto que tendría si así lo hiciera. La evaluación de riesgos es el proceso por el cual cada riesgo, y en especial riesgos de seguridad de la información, son caracterizados. [11]

Identificar amenazas y consecuencias requiere de un entendimiento del sistema en general y su implementación. Tanto el crecimiento en el número de tecnologías como el aumento de la complejidad aumentan la superficie de ataque y las vulnerabilidades del sistema, incrementando los riesgos. [11]

Como no es viable eliminar todos los riesgos de un sistema, debemos gestionarlos para que la energía y recursos ya invertidos en un sistema estén balanceadas contra el efecto de resultados no deseados. Este balanceo debe estar basado en una evaluación

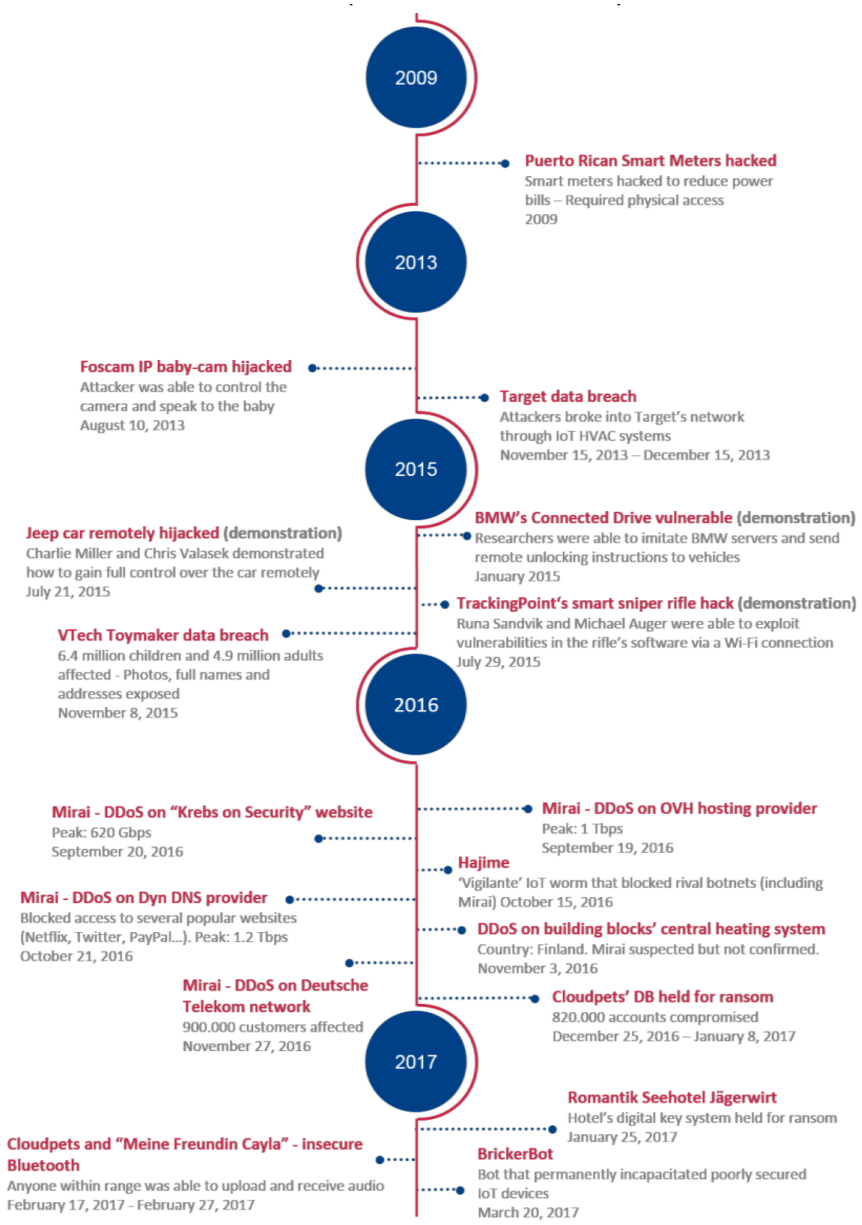

Figura 2.1: Línea de tiempo indicativa de incidentes de seguridad [2]

realista de las amenazas, los riesgos que generan y cómo pueden prevenir al sistema de cumplir sus funciones esperadas. [11]

Los ataques modernos abarcan un amplio rango de medios y motivos posibles. A continuación se presentan las amenazas en IoT en base a la taxonomía de ENISA.

Taxonomía de amenazas de ENISA

Siendo consistentes con la taxonomía de amenazas propuesta por ENISA en [13], la figura 2.2 muestra la taxonomía aplicada a IoT con algunos ejemplos de ataques listados. [2]

A continuación, también se listan y describen las distintas amenazas presentadas en dicha taxonomía en términos de IoT, clasificadas en distintas categorías en función del tipo de actividad que las origina [2]:

- Actividad maligna / Abusos
  - *Malware*
    Son programas de software diseñados para llevar a cabo acciones no deseadas o autorizadas en un sistema, sin el consentimiento del usuario, resultando en daño, corrupción o robo de información. Su impacto puede ser alto, y puede afectar tanto a los dispositivos como a la plataforma IoT.
  - *Kits de exploits*
    Es código diseñado para tomar ventaja de una vulnerabilidad con el fin de ganar acceso al sistema. Esta amenaza es difícil de detectar y en los ambientes IoT su impacto oscila entre alto y crítico, dependiendo de los recursos afectados. Tanto la infraestructura como los dispositivos IoT pueden ser blanco de esta amenaza.
  - *Ataques dirigidos*
    Son ataques diseñados para un objetivo específico, ejecutados durante un período largo de tiempo, y llevados a cabo en múltiples etapas. Mientras que el impacto de esta amenaza es mediano, detectar este tipo de ataques es usualmente muy difícil y lleva tiempo.
  - *DDoS*
    Se trata de un ataque donde múltiples sistemas atacan un objetivo único con el fin de saturarlo y hacer que no quede disponible. Se pueden realizar este tipo de ataques sobre los dispositivos IoT, la infraestructura o la plataforma del sistema.

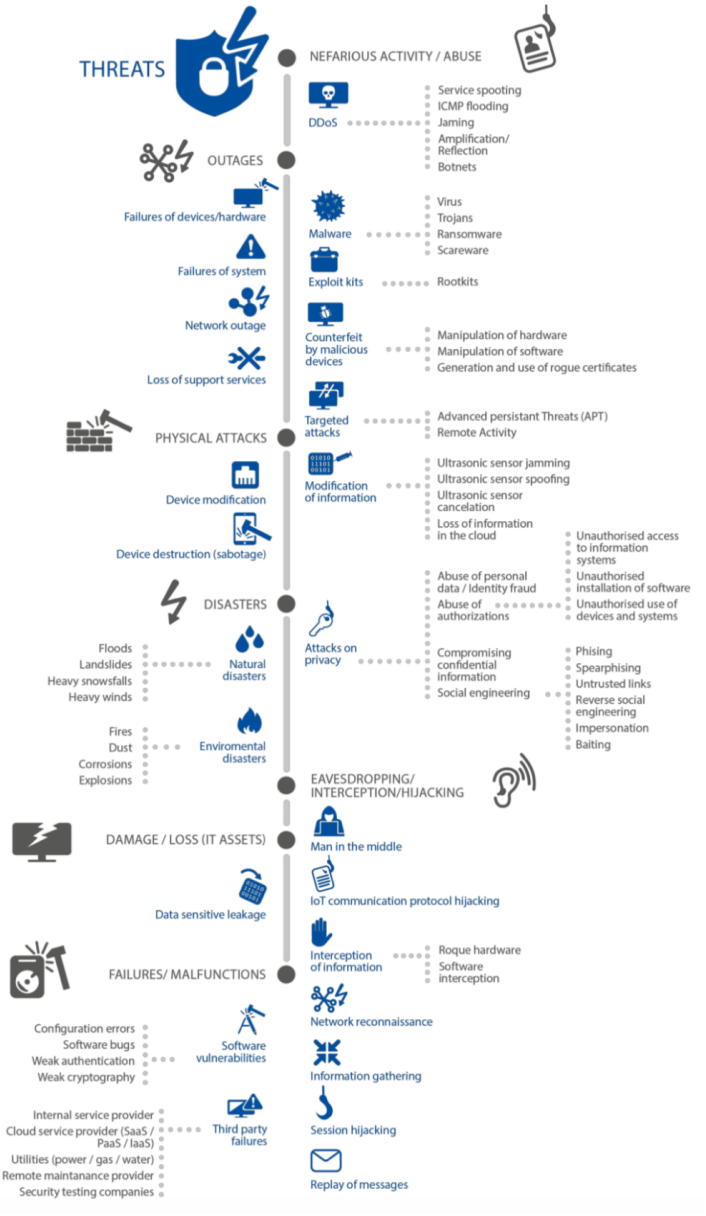

Figura 2.2: Taxonomía de amenazas en IoT [2]

- *Suplantación de identidad*
  Esta amenaza puede ser difícil de descubrir, dado que los dispositivos falsos no pueden ser fácilmente distinguidos de los originales. Estos dispositivos usualmente tienen backdoors y pueden ser usados para conducir ataques en otros sistemas.
- *Ataques a la privacidad*
  Esta amenaza afecta tanto la privacidad de los usuarios como la exposición de los elementos de la red a personas no autorizadas. Los dispositivos IoT, la información del sistema en sí misma, y la plataforma pueden ser afectados.
- *Modificación a la información*
  En este caso, el objetivo no es dañar el dispositivo, pero sí manipular la información con el fin de causar caos, o adquirir ganancias monetarias.

- Eavesdropping / Interceptación / Hijacking
  - *Man in the middle*
    Es un ataque donde se intercepta una comunicación y se lee el contenido. Todas las comunicaciones dentro de un sistema IoT pueden ser afectadas por esta amenaza.
  - *Hijacking del protocolo de comunicación IoT*
    En este ataque se toma control sobre una sesión de comunicación ya existente entre dos elementos de la red. El intruso es capaz de acceder a información sensible, incluyendo contraseñas. El hijacking puede llegar a usar técnicas agresivas como forzar la desconexión o la denegación de servicio.
  - *Reconocimiento de la red*
    Se obtiene pasivamente información interna sobre la red: dispositivos conectados, protocolos usados, puertos abiertos, servicios en uso, etc.
  - *Hijacking de sesiones*
    Se roban los datos de conexiones actuando como un host legítimo, con el fin de robar, modificar o borrar los datos transmitidos.
  - *Replay de mensajes*
    Este ataque usa una transmisión de datos válida de forma maligna al enviarla repetidamente o demorarla, con el objetivo de manipular o dar de baja el dispositivo objetivo.

- Daños / Pérdidas

- *Fuga de información sensible (leakage)*
  En esta amenaza datos sensibles son revelados, intencionalmente o no, a partes no autorizadas.

- Outages
  Se trata de la interrupción o falla en el suministro de la red, en los dispositivos y en los sistemas con los que se tienen contacto.

- Fallas / Mal funcionamiento
  Se incluyen vulnerabilidades del software de los elementos del sistema como fallas en terceros con los que se tiene relación directa.

- Desastres
  Se incluyen tanto desastres naturales como problemas en el ambiente donde se encuentra desplegado el equipamiento IoT.

- Ataques físicos
  Se trata de la alteración o destrucción de los dispositivos.

2.2.3. Situación actual y desafíos

Es importante analizar la situación actual en la que se encuentra el paradigma IoT con respecto a sus aspectos de seguridad, es decir, se necesita evaluar si los requerimientos se están satisfaciendo, si las amenazas y riesgos están siendo adecuadamente prevenidas y mitigadas, y si las contramedidas y acciones definidas son eficaces y suficientes.

Los autores de [2] realizan este análisis en base a lo que definen como "brechas" o "gaps" de seguridad. Los autores consideran la identificación y definición de estas brechas como parte crítica al abordar la ciberseguridad en IoT, y las definen como la diferencia entre el estado presente y el estado deseado. A continuación, se presentan las brechas identificadas por los autores:

- *Fragmentación en los enfoques y regulaciones de seguridad existentes*
  Es así que estamos ante una fragmentación en las regulaciones sobre la ciberseguridad en IoT, donde no existen imposiciones claras sobre medidas de seguridad y protocolos en todos los niveles, incluyendo los dispositivos, la red, etc. Será necesario entender a la ciberseguridad como una responsabilidad compartida entre todos los involucrados.

- *Falta de conciencia y conocimiento*

Existe una falta de conocimiento sobre IoT y sus riesgos de seguridad, tanto a nivel de los consumidores como de los expertos en tecnologías. Muchos incidentes de seguridad podrían ser evitados si los desarrolladores y fabricantes fueran conscientes de los riesgos con los cuales se enfrentan de forma diaria.

- *Diseño y/o desarrollo inseguro*
  El diseño mismo de un sistema IoT puede hacer de este un sistema inseguro por naturaleza, donde debido a cómo fue diseñado e implementado existen desde el principio vulnerabilidades de seguridad, que usualmente son solucionadas con un enfoque reactivo a medida que se presentan.

- *Falta de interoperabilidad a lo largo de los diferentes dispositivos IoT, plataformas y frameworks*
  Nos encontramos ante distintos modelos de seguridad, conceptos y taxonomías incompatibles. Esto requiere el desarrollo y uso de protocolos estandarizados que necesitan ser soportados por todos los fabricantes para asegurar un buen nivel de interoperabilidad con la mínima pérdida en eficiencia y seguridad.

- *Falta de incentivos económicos*
  Los principales fabricantes y comerciantes de IoT usualmente consideran a la funcionalidad y usabilidad como mucho más importante que implementar y diseñar seguridad. Por lo general los intereses económicos no están alineados con gastar en seguridad.

- *Falta de manejo adecuado del ciclo de vida de los productos*
  IoT comprende tal variedad de productos que, si se los deja sin atención, hace más vulnerable a toda la cadena de suministro tradicional. A través de su ciclo de vida, los dispositivos IoT deben ser capaces de ser reparados y actualizados rápidamente para asegurar su correcto comportamiento y solucionar todas las vulnerabilidades que son descubiertas constantemente.

2.2.4. Medidas y Técnicas

En esta sección se presentan medidas y técnicas de bajo nivel como forma de introducción a la gran cantidad de aspectos a considerar en cuanto a la seguridad de los componentes y del sistema se refiere.

El framework de seguridad presentado por los autores de [11] consta de la definición de distintos bloques funcionales que tienen determinados objetivos en cuanto a la seguridad del sistema se refiere. En cada uno de ellos se presentan consideraciones

y técnicas a tener en cuenta, a continuación se presentan las principales.

Protección de endpoints

Se considera como endpoint a cualquier elemento del sistema IoT que tiene capacidad de computación y comunicación y que expone capacidades funcionales. Las consideraciones de seguridad sugeridas por el framework son:

- *Arquitectura*
  Cada endpoint tiene distintas capacidades computacionales y de comunicación. Las decisiones tomadas por los fabricantes a la hora de implementar los endpoints determinan y tienen un efecto a largo plazo en la capacidad de seguridad de éstos. La seguridad de los endpoints puede implementarse tanto en el hardware como en el software.

- *Seguridad física*
  Algunos endpoints, como es el caso de los medidores inteligentes o el de sensores, no pueden encontrarse dentro de un perímetro de seguridad. Se vuelven necesarias distintas protecciones físicas aplicadas directamente que puedan proporcionar evidencia de alteraciones y que dejen expuestas las modificaciones realizadas.

- *Identidad*
  La determinación y asignación de una identidad es crucial a la hora generar y establecer confianza. Una gran cantidad de controles de seguridad dependen directamente de la identidad del endpoint y de los otros recursos y entidades con los cuales se comunica, y de la confianza que éstos establecen a través de dicha identidad. Tanto la autenticación como la autorización son ejemplos claros.

- *Control de acceso*
  El control de acceso se trata de verificar si una entidad, que solicita acceso a un recurso, se encuentra autorizada a hacerlo. La autorización depende de la verificación del mapeo entre la identidad de la entidad y los derechos y privilegios sobre los servicios y recursos, por lo que la autorización depende sobre la autenticación.

- *Integridad*
  Es necesario que la información referente a la identidad del endpoint sea apropiadamente asegurada y su integridad mantenida. A su vez, tanto los datos

almacenados como aquellos que se encuentran en movimiento y uso, deberán ser monitoreados y mantenidos de forma que la confianza en ellos no se vea aceptada.

- *Protección de datos*
  Los datos que son almacenados, utilizados y transferidos por el endpoint deberán ser protegidos utilizando distintas estrategias. La criptografía impone y fortalece la confidencialidad de los datos y además asegura su integridad. Puede ser usada en todos los datos, sólo en las porciones sensibles o en todo el medio de almacenado.

- *Monitoreo y análisis*
  Los mecanismos de monitoreo son de alta importancia y se encargan de la detección, por lo general en tiempo real, de anomalías en el sistema que podrían implicar ataques o que el sistema se encuentre comprometido.

- *Configuración y gestión*
  Una buena gestión del endpoint y su configuración es indispensable. El endpoint deberá proveer un control sobre los cambios que se realizan en su componentes, garantizando que éstos son seguros y no lo comprometerán.

- *Dispositivos con recursos limitados*
  Algo característico de los sistemas IoT es que, en general, los dispositivos que forman parte del sistema tienen recursos y capacidades limitadas. Nuevas técnicas de manejo del hardware serán necesarias para permitir a los dispositivos el cumplimiento de los requerimientos de seguridad (autenticación, protección en tiempo real, gestión de la configuración, entre otros).

Protección de comunicación y conectividad

Este bloque funcional busca proteger el tráfico de datos. Se utilizan las capacidades de autorización de los endpoints para implementar autenticación y autorización en el tráfico. Las consideraciones de seguridad sugeridas por el framework con respecto a esto son:

- *Protección criptográfica*
  - Políticas explícitas de comunicación entre los endpoints
  - Autenticación mutua entre endpoints criptográficamente fuerte
  - Mecanismos de autorización que impongan reglas de control de acceso derivadas de la política definida

    - Mecanismos criptográficos que aseguren la confidencialidad, integridad y novedad de la información intercambiada

- *Flujos de información*
  Se le denomina flujo de información a cualquier información "en movimiento", incluyendo mensajes IP, comunicaciones seriales, flujos de datos, señales de control, y otros. Se pueden prevenir ataques hacia ellos desarrollando medidas específicas.

- *Modelos y políticas de seguridad*
  Un modelo de seguridad es aquel que define las relaciones permitidas y prohibidas entre distintos sujetos y objetos del sistema, capturando así, tanto de forma formal como informal, las políticas de seguridad del sistema. Estas políticas deberán ser evaluadas constantemente, para garantizar su correctitud y completitud.

Monitoreo y análisis de seguridad

Es necesario que el estado del sistema sea preservado a lo largo del ciclo de vida operacional. Este bloque funcional entonces tiene como responsabilidad la recolección de datos sobre el estado general del sistema, analizando eventos pasados y actuales, que permitan la detección de posibles violaciones a la seguridad o amenazas y la predicción de riesgos futuros. La política de seguridad del sistema deberá indicar luego las acciones a tomar en cada caso, para las cuales existe un gran rango de posibilidades.

Gestión de la configuración de seguridad

Este bloque funcional es responsable de controlar los cambios tanto en las funcionalidades operacionales como en los controles y mecanismos de seguridad que garantizan la protección del sistema en sí misma. Se busca que todo cambio sea realizado de forma segura, controlada y confiable, brindando estabilidad al sistema.

2.3. Smart Grids

2.3.1. Definiciones

El grupo de organismos reguladores europeos de la electricidad y el gas define Smart Grid como [14]:

"Una red eléctrica que puede integrar el comportamiento y las acciones de todos los usuarios conectados en una forma eficiente en cuanto a costos, incluyendo productores, consumidores y actores que son tanto productores como consumidores, para asegurar una red eléctrica que haga un uso más eficiente de los recursos con un costo inferior, asegurando menores pérdidas y mayor calidad y seguridad de la demanda.".

Para el National Institute of Standards and Technology (NIST) [15]:

"Una Smart Grid es una red moderna con flujo bidireccional de energía que provee, mediante comunicaciones bidireccionales y capacidades de control, una serie de funcionalidades y aplicaciones a la red".

El informe del Electric Power Research Institute (EPRI) adopta la siguiente definición, por la cual el concepto de Smart Grid hace referencia a [16]:

"La modernización del suministro eléctrico para lo cual se debe monitorear, proteger y optimizar automáticamente la operación de los elementos interconectados a ella, desde la generación (tanto si está centralizada como distribuida), pasando por la red de transporte y distribución hasta los consumidores industriales, sistemas automáticos en edificios, consumidores finales y sus equipos (vehículo eléctrico, electrodomésticos, etc.)".

En base a todas estas definiciones podemos sacar algunas características en común de las Smart Grid. Basada en una red de comunicación y control de los actores involucrados, la producción de energía eléctrica debe ser coordinada y demandada en una forma más efectiva. En términos generales, una Smart Grid provee una infraestructura de control y comunicación que permite la interacción entre los participantes de la red eléctrica. En particular, las dos aplicaciones principales son el monitoreo y el control avanzado de la red de distribución. [17]

A grandes rasgos, según se muestra en [18], las redes eléctricas inteligentes se diferencian de las actuales en los siguientes aspectos:

- Las redes actuales consisten en una mezcla de sistemas electromecánicos y

sistemas digitales, mientras que en un futuro se pasará a contar con sistemas en su mayoría digitales.

- La comunicación actualmente es unidireccional, mientras que se está transicionando hacia una comunicación bidireccional.
- También habrá cambios en la generación de energía, pasando de un modelo de generación centralizado a una combinación entre generación centralizada y distribuida.
- Actualmente se cuenta con un número limitado de sensores pero en un futuro se pasará a contar con una gran cantidad de sensores, teniendo una red completamente monitoreada, con mayores niveles de control y supervisión.
- Hoy en día no se cuenta con muchos datos y por lo tanto no hay sistemas de gestión de datos. Con el despliegue de los medidores inteligentes, se contará con una cantidad muy grande de datos, generando la necesidad de contar con sistemas de almacenamiento y de tomar medidas para garantizar la protección de datos sensibles de los clientes. A su vez, mediante el uso de los dispositivos inteligentes, el consumidor pasa a tener un rol más activo en lugar de ser simplemente un consumidor pasivo.
- La facturación pasará a ser más compleja, pudiendo cambiar las tarifas en forma dinámica de acuerdo a la demanda en tiempo real o incluso según la calidad del servicio.
- En las redes eléctricas inteligentes será posible identificar los problemas de calidad de servicio y solucionarlos rápidamente.

Estas diferencias o mejoras que supondrá la implementación de Smart Grids pueden verse en mayor detalle en la figura 2.3.

Ha habido grandes esfuerzos por lograr la estandarización del concepto de Smart Grid, principalmente en Europa. Para esto se ha trabajado en la definición de estándares, de manera de lograr consenso en cuanto a las funcionalidades a ofrecer, interfaces, protocolos y otros estándares a utilizar. Además, destacan los modelos de arquitectura de referencia desarrollados por el NIST [17] y por el Smart Grid Coordination Group [19].

2.3.2. AMI (Advanced Metering Infrastructure)

Las AMI son un componente crucial de las Smart Grids, se trata de la infraestructura de medición. Maneja la comunicación bidireccional entre los medidores

Red actual	Red del futuro
Electromecánica-Digital	100% Digital
Comunicación unidireccional	Comunicación bidireccional
Generación centralizada	Combinación de generación centralizada y distribuida
Topología mallada en redes de MT y BT[62] con carácter general. Explotación radial	Posible explotación de topología mallada.
Separación contable, jurídica y funcional de las actividades	Reglas más detalladas de separación de actividades y nuevos operadores (generadores puros, generadores virtuales, almacenadores, comercializadores puros, clientes/generadores)
Número limitado de sensores en MT y BT	Red de MT monitorizada y con gran número de sensores
Escaso volumen de datos en BT. Ausencia de sistemas de gestión de datos.	Necesidad tanto de sistemas de almacenamiento, como de protección y seguridad de una ingente cantidad de datos
Red con escasa visibilidad en BT	Red con mayores niveles de control y supervisión en MT y BT
Herramientas de ayuda a la reposición	Reposición semiautomática o automática
Decisiones soportadas por el sistema en AT y MT	Ampliación a la BT
Consumidores pasivos	Consumidores activos
Customer relationship management (CRM) simplificados	CRM avanzados, facturadores más complejos, *Business Intelligence*
Calidad de suministro. Persisten problemas de huecos de tensión, perturbaciones, armónicos, etc.	Calidad eléctrica que satisface a los consumidores, incluidos los industriales. Identificación y resolución de problemas de calidad eléctrica. Varios tipos de tarifas para varios tipos de calidades eléctricas
Ausencia de servicios al sistema de distribución	Participación activa de los recursos energéticos y del distribuidor en los nuevos servicios al sistema de distribución

Figura 2.3: Tabla comparativa: Cambios en la infraestructura eléctrica [18]

inteligentes y los sistemas, permitiendo enviar, recibir y procesar los datos de consumo de los clientes, además de operaciones adicionales sobre la red.

Según [20] las arquitecturas más comunes para este tipo de sistemas incluyen los siguientes componentes:

- *Medidores inteligentes (Smart Meters)*
  Dispositivos encargados de medir y reportar el consumo de energía.
- *Concentradores (Data Concentrators)*
  Dispositivos que procesan datos de varios medidores.
- *Head End System (HES)*
  Sistema encargado de recolectar los datos, procesarlos y tomar acciones en base a los mismos.
- *Redes de area local (Home Area Network, Neighborhood Area Network)*
  Permiten la comunicación bidireccional entre los medidores y los concentrados.
- *Wide Area Network (WAN)*
  Permite la comunicación bidireccional entre los concentradores y el HES o entre medidores y HES en algunos casos.

Los medidores inteligentes son una pieza central dentro de las AMI. Ellos son los encargados de medir el consumo de energía de los consumidores finales y de transmitir esta información al proveedor del servicio para que pueda facturar al consumidor en forma automática. Además, se encargan de la medición de otros parámetros técnicos de interés para el proveedor para balancear la carga en la red de energía eléctrica, y también permiten la desconexión o reconexión en forma automática de clientes, tanto por razones contractuales, como por ejemplo que no se encuentren al día con el pago del servicio, como por razones técnicas o de seguridad debido a problemas en la red. [20]

Según [21], los sistemas de Smart Meters deben al menos soportar las siguientes funciones:

- Adquisición de información sobre la red local doméstica (HAN, Home Area Network), y procesamiento y comunicación de la misma sobre una Wide Area Network (WAN) a la empresa.
- Almacenamiento de la información de consumo y demanda. Comunicación del consumo en tiempo real a la empresa para su facturación y contabilidad.
- Comunicación al usuario del consumo en tiempo real, por ejemplo a través de un display en el dispositivo o a través de una aplicación móvil.

- Comunicación bidireccional con el HES, permitiendo:
  - Control del medidor sin requerir acceso físico al mismo.
  - Descarga/Actualización del software en el medidor de manera remota.
- Escalabilidad e interoperabilidad, con el objetivo de soportar varios proveedores de servicio. Esto resulta de importancia en mercados como el europeo, donde la comercialización de energía eléctrica no está restringida a un único proveedor.
- Construcción de perfiles de carga.
- La privacidad y seguridad de los datos de los consumidores debe estar garantizada mediante la utilización de servicios y mecanismos de seguridad apropiados.

Los medidores inteligentes se instalan comúnmente en los hogares de los consumidores, y típicamente se conectan a un concentrador, en donde se agrupa la información de todos los medidores conectados a él. Estos concentradores pueden ser a nivel de edificio, de cuadra o grupo de hogares o a nivel de barrio incluso. Otra solución posible es que los medidores se comuniquen directamente en forma individual con el sistema de la empresa proveedora del servicio (HES, Head End System). [22]

Las comunicaciones entre medidores y concentradores, y entre concentradores o medidores y el HES, se puede realizar sobre distintos tipos de red, siendo muy común que se utilice la tecnología PLC (Power Line Communication) utilizando la misma red eléctrica para conectar medidores y concentradores, y tecnologías wireless como por ejemplo GPRS/UMTS para la conexión con el HES. [22]

Un ejemplo de arquitectura posible puede observarse en la figura 2.4.

2.3.3. Desafíos de seguridad y privacidad

El desarrollo de las Smart Grids genera varios desafíos de seguridad agravados por la escala y la complejidad que toman estos sistemas. Muchos subsistemas que se encontraban previamente aislados ahora requieren interoperabilidad y comunicación constante entre ellos.

Del lado de los proveedores, preocupa la presencia de vulnerabilidades de seguridad en los dispositivos disponibles en el mercado que permita que potencialmente un atacante pueda modificar las lecturas de los medidores, provocando grandes pérdi-

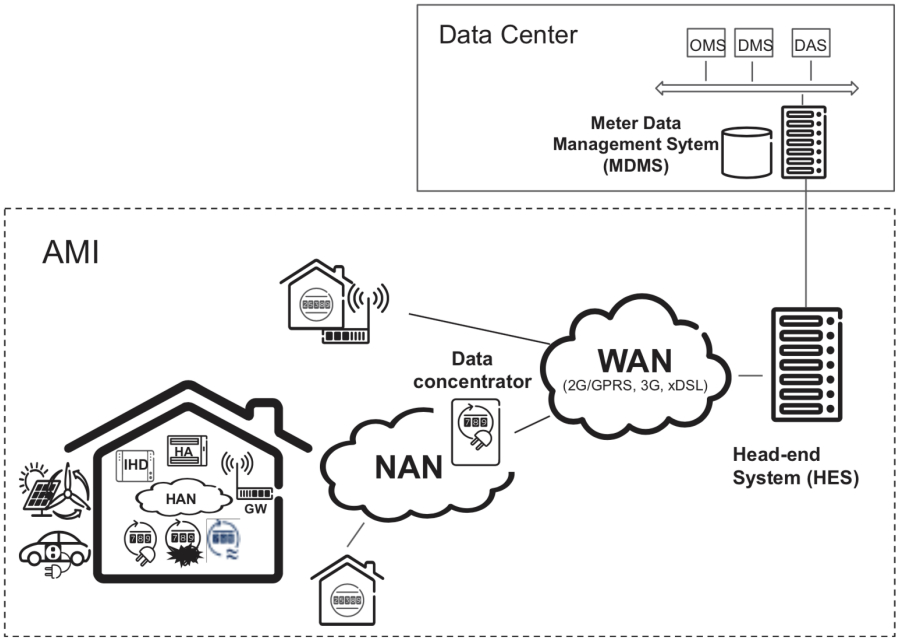

Figura 2.4: Red de medidas inteligentes [20]

das económicas, tanto a los consumidores como a los proveedores. Además, muchos de estos dispositivos cuentan con la capacidad de ser apagados en forma remota, pudiendo resultar en graves problemas de disponibilidad del servicio de ser atacados con malas intenciones. [17]

Por otra parte, del lado de los consumidores, preocupa que la información sobre el consumo de los hogares caiga en manos equivocadas o incluso que se haga un mal uso de la misma por las empresas proveedoras del servicio. Esta información es de carácter sensible ya que al tener datos del consumo en tiempo real se puede hacer un análisis de los hábitos de los consumidores, incluso llegando al detalle de saber si se encuentran personas o no en el hogar, y qué actividades realizan y en qué horarios. Esta información inferida puede ser utilizada con varios fines, como por ejemplo publicidad dirigida, estudio de comportamiento de consumidores, robo, etc. En [23] se muestra cómo a través de la información recolectada se puede extrapolar un perfil de usuario muy preciso.

Las soluciones de seguridad existentes no siempre pueden ser aplicadas directamente, debido a que las Smart Grid son sistemas ciberfisicos con problemáticas particulares distintas a los de los sistemas puramente computacionales. [17]

Además, en la realidad de las redes eléctricas, las medidas adicionales de seguridad pueden llegar a ser contraproductivas, generando que una señal de vital importancia no llegue a destino a tiempo debido a medidas como chequeos de in-

tegridad y autenticación. Las medidas que se tomen para garantizar la seguridad tienen que estar bien justificadas y enfocadas correctamente. [17]

2.3.4. Ataques posibles

Con el desarrollo de Smart Grids, además de los ataques físicos a la red, se suman ataques a las redes de comunicación o a los sistemas informáticos de control, los cuales pueden tener un impacto negativo sobre la red eléctrica, tanto intencionalmente o como efecto secundario. [17]

Los ataques son posibles debido a la existencia de vulnerabilidades y, debido a la inexperiencia con este tipo de tecnologías, en este campo es muy probable que se lancen al mercado productos que no han pasado por controles estrictos de aseguramiento de calidad. [17]

En [21] se describen los siguientes tipos de ataque:

- *Eavesdropping (Escuchas Pasivas)*
  Se trata de un ataque pasivo en el que el atacante escucha en la WAN la información en tránsito desde el medidor hasta el HES. Estos ataques afectan la privacidad de los consumidores y tienen distintas posibles consecuencias que ya fueron discutidas previamente.

- *Ataques de denegación de servicio (DoS)*
  Estos ataques pueden tener como objetivo la red eléctrica en su totalidad o partes de la misma, buscando dejar no disponible el servicio. Se pueden realizar enviando una cantidad de mensajes mucho mayor a la normal tanto a los medidores como al HES.

- *Inyección de paquetes*
  Al inyectar mensajes falsos en la comunicación, se puede enviar información de facturación falsa, generando costos económicos a los consumidores y/o a los proveedores, o incluso se puede enviar mensajes para desconectar a un hogar de la red.

- *Inyección de malware*
  Mediante la inyección de malware en la red, se puede afectar la comunicación entre los dispositivos y comprometer los procesos de facturación y reporte. En las redes eléctricas esto puede provocar el mal funcionamiento de la infraestructura de distribución.

- *Conectar o desconectar dispositivos en forma remota*
  Al enviar mensajes de desconexión a los medidores se puede lograr la desconexión de uno, varios o incluso de toda la red de medidores.

- *Manipulación del firmware*
  Las manipulaciones al firmware pueden ser tanto a la parte encargada de medir el consumo como a otras partes del mismo. Este ataque puede ser realizado teniendo acceso físico al medidor pero también si está habilitada la actualización del firmware de forma remota.

- *Ataques Man-in-the-Middle (MitM)*
  Pueden ser realizados tanto en las redes locales de medidores, es decir, entre medidores y concentrador, o en la WAN pudiendo manipular todo el tráfico en tránsito hacia y desde el HES. Estos ataques pueden tener consecuencias muy graves, incluso comprometiendo la seguridad nacional.

Como se puede observar, las consecuencias de estos ataques van desde la divulgación de información de los consumidores, afectando su privacidad, lo cual puede traer consecuencias legales para las empresas, hasta problemas de seguridad nacional.

Frente a este escenario resulta apremiante tomar medidas para evitar o mitigar el riesgo de sufrir estos ataques. Diferentes contramedidas pueden ser aplicadas, algunas son familiares porque se tratan de las mismas aplicadas a las redes de propósito general, como por ejemplo [21]:

- *Comunicaciones cifradas*
  Se recomienda doble cifrado, tanto en la capa de aplicación, asegurando cifrado end-to-end, como en la capa de transporte, utilizando protocolos ya existentes como por ejemplo TLS.

- *Protección de integridad*
  Es de vital importancia garantizar la integridad de los mensajes, utilizando por ejemplo Message Authentication Codes (MAC) para asegurar la integridad de la información de consumo energético.

- *Verificación de autenticidad*
  Tanto los participantes en la comunicación como la información transmitida deben ser autenticados. Es deseable la implementación de autenticación mutua.

- *Utilización de Gateways*
  Esta es una solución propuesta por algunos países europeos. Consiste en la utilización de un dispositivo que actúe como gateway entre los dispositivos

de medición y el HES. El gateway recibe los datos de consumo directamente de los medidores y concentradores y comunica esta información al HES en forma periódica, cada cierto intervalo de tiempo. Este nuevo dispositivo es el responsable de garantizar la privacidad de los consumidores.

- *Sistemas de detección y prevención de intrusiones*
  Este tipo de sistemas ayudan a detectar nodos que estén actuando como fuentes de ataques y excluirlos de la red.

2.4. DLMS/COSEM

DLMS/COSEM es un protocolo de capa de aplicación para intercambio de medidas de energía desde y hacia medidores inteligentes, soportando aplicaciones como lectura de medidas en forma remota, control remoto de dispositivos de medida y servicios de valor agregado de medición de cualquier tipo de energía. Este protocolo se describe en una serie de estándares que fue adoptado por la International Electrotechnical Commission (IEC) como la serie de estándares IEC 62056. Estos estándares han sido adoptados por un gran número de fabricantes y proveedores de servicio, convirtiéndolo en uno de los más implementados en los medidores inteligentes y en particular, el estándar utilizado en la solución de UTE.

A continuación, describiremos brevemente el protocolo y pasaremos a presentar sus vulnerabilidades conocidas estudiadas, tanto del protocolo en sí como posibles vulnerabilidades de implementación.

2.4.1. Descripción del protocolo

El protocolo DLMS/COSEM, utilizando un enfoque basado en objetos, provee distintas clases de interfaz para representar objetos de la realidad de las AMI y sus funcionalidades. Mediante la instanciación de estas clases, se representa el estado y la funcionalidad del equipamiento de medición, de forma de exponerlo a través de la red de comunicación.

Los dispositivos de la AMI se modelan en el protocolo como dispositivos físicos, los cuales a su vez contienen uno o varios dispositivos lógicos que se encargan de modelar funcionalidades específicas de cada equipo. Es en estos últimos donde se encuentran los objetos que modelan la funcionalidad ofrecida por los dispositivos. Estos objetos, al igual que en el paradigma de programación orientada a objetos, no

son más que una agrupación de atributos y métodos. [24]

Los medidores actúan como servidores, los cuales son consultados por aplicaciones cliente que obtienen datos, proveen información de control o ejecutan acciones en el medidor a través de los atributos y métodos expuestos en los objetos definidos en el medidor.

Para definir un medidor basta con definir un dispositivo físico con sus dispositivos lógicos e instanciar las clases de interfaz deseadas. El estándar define 70 clases de interfaz que pueden ser utilizadas, cada una con distintos atributos y métodos disponibles. Por ejemplo, para definir una medida de energía se provee una clase Register, la cual será instanciada por cada uno de los tipos de medidas que mantenga el medidor. Tomando como ejemplo un medidor que se encargue de medir el consumo de energía eléctrica, de gas y de agua, se definirán tres dispositivos lógicos, uno por cada tipo de energía, y dentro de cada uno de ellos se tendrá una instancia de la clase Register, la cuál contendrá un atributo con el valor de la medida del consumo de energía. [24]

2.4.2. Acceso a los objetos

Para poder acceder a los objetos en un servidor DLMS/COSEM, es necesario establecer una Application Association (AA) con el cliente, de manera de identificar a los participantes y establecer el contexto en el cual se llevará a cabo la comunicación, proveyendo por ejemplo cuál mecanismo de autenticación debe ser utilizado. Los servidores siempre poseen una instancia de un tipo especial de clase llamada Association, que contiene esta información y además contiene una lista de todos los objetos que son accesibles en ese servidor en particular para que una aplicación cliente pueda saber a qué información puede acceder y cómo hacerlo. [24]

El protocolo provee dos formas de acceder a los objetos, por Logical Name (LN) o por Short Name (SN).

Cada objeto tiene siempre un LN, en general los fabricantes usan los mismos valores en sus dispositivos, de forma que una misma aplicación encargada de recolectar datos de medidores pueda hacerlo sin importar el fabricante. El LN es el primer atributo de un objeto COSEM y junto con el id de la clase de interfaz define el significado del objeto. El LN se define como un código OBIS (Object Identification System), de 6 bytes, por ejemplo el código del objeto Association es siempre 0.0.40.0.0.255, y dentro de cada objeto los atributos y métodos se identifican con un

id numérico. [24]

En la forma de acceso por LN, los métodos y atributos se acceden mediante el id de la clase de interfaz, el valor del LN y el índice del atributo o del método al que se quiere acceder. De la siguiente manera: id de la clase | logical name | id atributo o método. [24]

En cambio en la forma de acceso por SN, cada objeto es mapeado a un SN. Esta es una forma de acceso simplificada, indicada para ser utilizada por dispositivos simples. En este caso, cada atributo y método de los objetos es identificado con un entero de 13 bits. [24]

Mediante estas dos formas el cliente puede acceder a los objetos, pudiendo leer o modificar los valores de sus atributos o invocando acciones mediante sus métodos.

2.4.3. Autenticación y control de acceso

Se definen tres niveles de autenticación en el protocolo [24]:

- *Lowest Level Security*
  Ni el cliente ni el servidor son autenticados.
- *Low Level Security (LLS)*
  Solo el cliente se autentica presentando un password al servidor. Este tipo de autenticación solo debe ser utilizado cuando la comunicación se lleva a cabo sobre un canal seguro, de manera de evitar escuchas y replay de mensajes.
- *High Level Security (HLS)*
  Se utiliza autenticación mutua, tanto el cliente como el servidor se autentican contra su contraparte. Este es el método de autenticación recomendado ya que en general no se puede garantizar la seguridad del canal de comunicación. Se puede optar por HLS-MD5 (Message Digest 5), HLS-SHA1 (Secure Hash Algorithm 1) o HLS GMAC (Galois Message Authentication Code). [25]

Hay muchos mecanismos posibles para controlar el acceso a los objetos COSEM, el más simple es utilizando la dirección de los clientes para definir a qué objetos ese cliente en particular puede tener acceso y de qué tipo (lectura o escritura). [24]

2.4.4. Establecimiento de la conexión

Para el manejo de la conexión entre un cliente y un servidor DLMS/COSEM, se utilizan los servicios provistos por Association Control Service Element (ACSE). Se cuenta con 4 tipos de mensajes para establecer y terminar una conexión, estos son [24]:

- AARQ (Association Request)
- AARE (Association Response)
- RLRQ (Release Request)
- RLRE (Release Response)

El formato de estos mensajes se encuentra definido en el estándar OSI/IEC 8650-1 y se puede consultar en el apéndice **??**.

Al comenzar una conexión, el cliente envía un mensaje AARQ al servidor, indicando algunos de sus datos y qué método de cifrado y autenticación desea utilizar, además se envía información adicional como qué versión del protocolo utilizar. Como respuesta a este mensaje, el servidor envía un mensaje AARE, aceptando o rechazando el intento de conexión, o si se utiliza HLS como método de autenticación, aceptando parcialmente la conexión, esperando a que se complete el paso extra de autenticación mutua. Este paso extra consiste en la respuesta a un desafío presentado por la contraparte, el servidor le envía al cliente un nonce y el cliente le envía al servidor su respuesta en base a este nonce, y viceversa. En el caso de utilizar MD5 o SHA1, las respuestas HLS se calculan como f(nonce | secreto compartido). Al utilizar GMAC para calcular la respuesta al desafío, se aplica GMAC a la concatenación de un byte de control 0x10, la clave de autenticación y el nonce. [25]

Luego de establecida la conexión el cliente puede enviarle al servidor requests, tanto para obtener datos como para invocar métodos, para las cuáles recibirá respuestas del servidor. Los mensajes cuentan con un header que indica el tipo de mensaje que es y el tipo de comunicación, los valores posibles pueden verse en la figura 2.5:

Finalmente, al querer terminar una conexión, el cliente le envía al servidor un mensaje RLRQ solicitando la finalización y luego el servidor responde con un mensaje RLRE confirmando la conclusión de la comunicación entre ambos. Este intercambio, desde comienzo a fin de la conexión, puede verse en la figura 2.6.

Message	Plaintext	Global	Dedicated
Get Request	192	200	208
Set Request	193	201	209
Event Notification Request	194	202	210
Action Request	195	203	211
Get Response	196	204	212
Set Response	197	205	213
Action Response	198	206	214

Figura 2.5: Tipos de mensaje DLMS/COSEM [25]

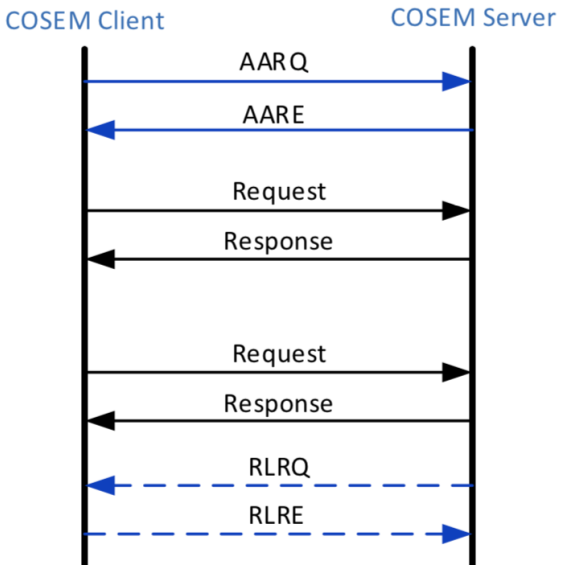

Figura 2.6: Flujo de comunicación entre cliente y servidor [24]

2.4.5. Perfiles de comunicación

El protocolo DLMS/COSEM permite la comunicación tanto sobre redes TCP-UDP/IP como HDLC. En el caso de las redes TCP-UDP/IP, los servicios de COSEM son soportados por la capa de transporte COSEM, que consiste en un wrapper sobre el protocolo TCP o UDP que agrega un header de 8 bytes antes del mensaje, indicando la versión del protocolo utilizada, los puertos de origen y destino, y el largo del mensaje enviado. De esta forma, se asegura que se reciba todo el mensaje antes de ser procesado. [24]

2.4.6. Uso de criptografía

Cada dispositivo cuenta con tres claves [25]:

- *Clave maestra*
  Se trata de una clave AES, programada en el firmware del medidor al momento de fabricación. Es requerida para cambiar el método de cifrado y las claves de cifrado y autenticación.

- *Clave de cifrado*
  Clave de 128 o 256 bits, utilizada para el cifrado de los mensajes DLMS/COSEM.

- *Clave de autenticación*
  Es una clave del mismo largo que la clave de cifrado. Se utiliza para calcular el tag de autenticación de los mensajes enviados, este tag de autenticación sirve como prueba de la integridad del mensaje y demuestra el conocimiento de la clave de autenticación por parte del emisor del mensaje.

Hay tres modos de comunicación posible: en texto plano, es decir, sin cifrar, cifrada utilizando la clave de cifrado global o cifrada utilizando una clave dedicada para la sesión en curso. [25]

Para el cifrado de los mensajes el protocolo utiliza el modo de cifrado en bloque Galois Counter Mode (GCM) y AES-128. Cada bloque del keystream se calcula usando AES-128 en base a la clave de AES de 128 bits, un vector de inicialización (IV) único y el contador de bloque, el cual se expresa como un entero sin signo de 32 bits que comienza con el valor 1. Es muy importante que el IV sea único, ya que repetir el keystream puede tener consecuencias graves, comprometiendo la confidencialidad de la comunicación. El IV en teoría puede ser de cualquier largo, pero IVs de largo menor a 96 bits se consideran inseguros. DLMS/COSEM sigue esta recomendación y utiliza como IV la concatenación del AP Title, valor que identifica a cada dispositivo y es intercambiado durante el establecimiento de la conexión, de 64 bits, seguido del frame counter, de 32 bits. [25]

Sea Kn el bloque n del keystream, tenemos que Kn = EncryptAES(IV | Contador). El primer bloque se reserva para cifrar el tag de autenticación y el resto de los bloques para el payload. Si el número de bytes en el payload a cifrar no es divisible por el tamaño del bloque, que en este caso es 128 bits, entonces se agregan bits de padding al último bloque del mismo. [25]

En las figura 2.7 y 2.8 se puede observar un diagrama del funcionamiento del algoritmo de cifrado y descifrado, respectivamente.

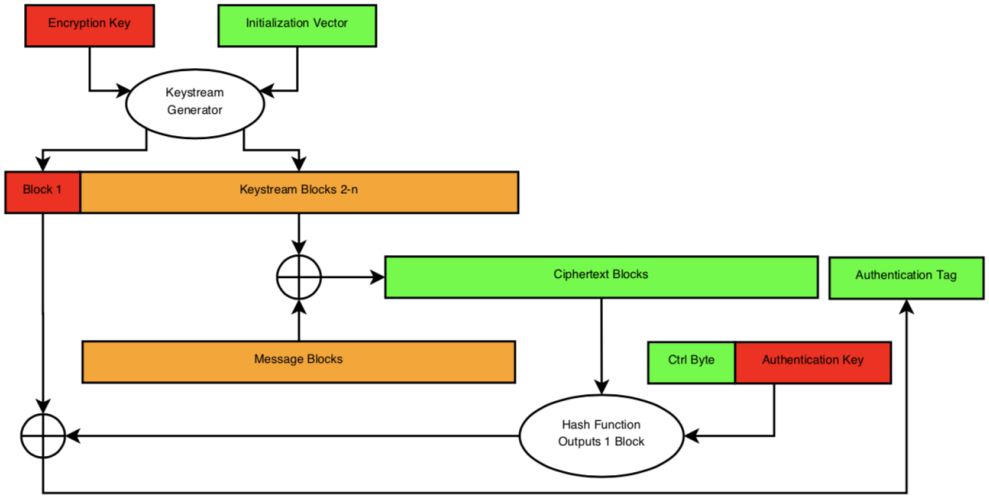

Figura 2.7: Algoritmo de cifrado [24]

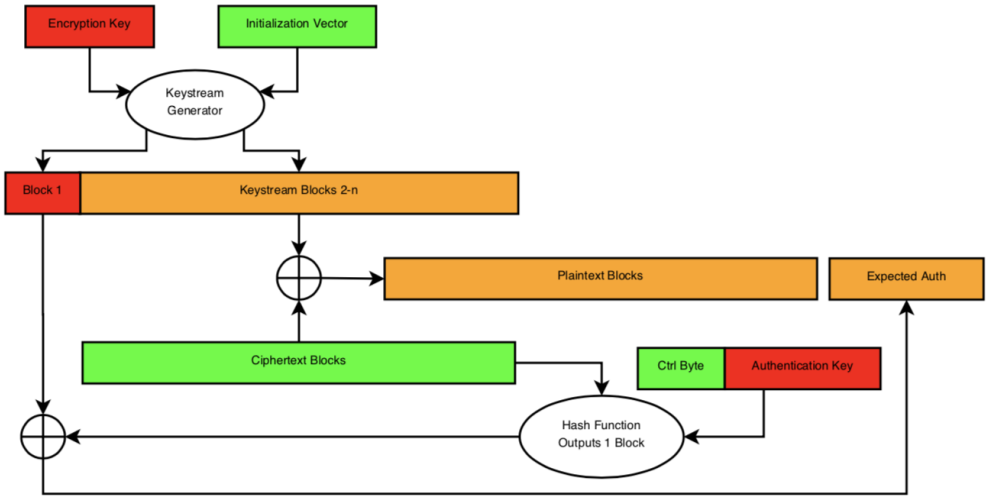

Figura 2.8: Algoritmo de descifrado [24]

2.4.7. Vulnerabilidades

El protocolo DLMS/COSEM no está libre de vulnerabilidades, tanto del protocolo en sí como particulares a sus diferentes implementaciones. A continuación mostraremos algunas de las vulnerabilidades conocidas estudiadas en [25].

Vulnerabilidades del protocolo

- *Autenticación opcional*
  El uso de autenticación es opcional según la definición del protocolo, y el mismo es independiente del cifrado de los mensajes. Un atacante puede manipular el byte de seguridad de los mensajes y truncar el mensaje de modo de eliminar el byte de autenticación, y así mantener una comunicación sin autenticación, sin que el receptor pueda saber si esta era la intención original del emisor.

- *Filtrado de información*
  Como vimos, cada mensaje contiene un header indicando el tipo de mensaje y modo de comunicación utilizado. Esto es innecesario, ya que se revela el tipo de mensaje incluso al tratarse de una comunicación cifrada, haciendo posible que atacantes sepan qué información se está transmitiendo. Esto podría ser sustituido simplemente indicando si se trata de comunicación cifrada utilizando la clave global o una clave dedicada y el tipo de mensaje indicado cifrado con el resto del mensaje. Los dispositivos no requieren esta información adicional para poder descifrar el mensaje.

- *Métodos de autenticación vulnerables*
  En HLS, el método que se utiliza para autenticar al cliente y al servidor es el mismo. Dadas las circunstancias adecuadas, un atacante podría impersonar a un servidor válido mediante un ataque de replay del AP Title, nonce y la respuesta al desafío de autenticación. Además, ataque de diccionario offline son posibles, dado que se conoce el nonce (enviado en texto plano), la respuesta al desafío y la función de autenticación. Con esta información, dado suficiente tiempo y recursos, la clave utilizada puede ser calculada.

- *Posibilidad de inyección de respuestas al cliente*
  Las respuestas no están ligadas a las requests, es decir, dado una request enviada por el cliente las respuestas pueden ser reemplazadas por otro mensaje y, siempre y cuando contenga el tipo de datos esperado, será tomado como una respuesta válida.

Vulnerabilidades de implementación

A continuación se listan algunas vulnerabilidades que pueden estar presentes en implementaciones particulares del protocolo [25]:

- *Incumplimiento del uso del frame counter*
  El servidor acepta mensajes cuyo frame counter sea menor o igual al contador del último mensaje cifrado recibido de un dispositivo. Esto resulta en una falta de protección adecuada ante ataques de replay.

- *Uso de nonces predecibles*
  Si se utilizan nonces predecibles un atacante podría calcular las respuestas HLS necesarias para lograr autenticarse con el servidor o cliente.

- *Se permiten AP Titles idénticos*
  Si se permiten comunicaciones entre dos dispositivos con el mismo AP title, un cliente o servidor puede permitir la comunicación con un dispositivo DLMS /COSEM con su mismo AP title. Esto podría permitir ataques de replay.

- *Se permiten mensajes cifrados sin autenticación*
  Los mensajes cifrados sin autenticación son susceptibles a manipulación, por lo que es deseable que no sean procesados.

- *Se permiten métodos de autenticación inseguros*
  LLS no debería ser utilizado ya que la password se envía en texto plano. Además, métodos de autenticación propietarios, y las versiones MD5 y SHA1 de HLS son vulnerables a ataques man-in-the-middle.

- *Los mensajes son procesados únicamente en base al header que indica el tipo de mensaje, sin verificar el contenido de los mismos*
  Si se procesan mensajes solo teniendo en cuenta el header indicando el tipo de mensaje, un atacante podría enviar un mensaje cuyo tipo indique que se trata de un mensaje cifrado, pero con los bits de cifrado y autenticación desactivados en el byte de seguridad, potencialmente engañando al servidor para que procese un mensaje malicioso enviado en texto plano.

- *Se envían mensajes AARE cifrados en respuesta a AARQ en texto plano*
  Esto puede permitir utilizar esta información para obtener el keystream, ya que los valores de los mensajes AARE son conocidos en base al AARQ enviado.

- *Posibilidad de ataques de diccionario online*
  Se permiten múltiples intentos de autenticación, permitiendo ataques de dic-

cionario online. Luego de N intentos de conexión, los mensajes de error deberían ser los mismos.

- *Se permiten mensajes con un tag de autenticación inválido*
  Si se procesan mensajes sin controlar el tag de autenticación los mensajes son vulnerables a violaciones de integridad.

- *Se permiten AP Titles arbitrarios*
  Si se permiten conexiones desde cualquier cliente, en lugar de permitir conexiones solamente de clientes con ciertos AP Titles esperados, un atacante podría llevar a cabo un ataque de Denial of Service (DoS), iniciando conexiones desde una cantidad suficiente de clientes.

Capítulo 3

Análisis

En la presente sección se describe el trabajo realizado referente a la definición y análisis de los requerimientos de la herramienta, de forma de evaluar la viabilidad de la misma y aproximarnos de forma gradual a su implementación. Diversos obstáculos e interrogantes enfrentados fueron provocando cambios en la idea, generando que la misma fuese evolucionando, y con ella la visión sobre cómo implementarla.

Como resultado de esta etapa, se decidió implementar una herramienta que permita ejecutar un diagnóstico de la implementación del protocolo DLMS/COSEM en medidores y/o concentradores, mediante la ejecución de pruebas de vulnerabilidades conocidas y de ataques ya implementados en la herramienta. A su vez, de forma de brindar una mayor libertad al usuario, se definió que la herramienta sea fácilmente extensible y permita la implementación de nuevas pruebas y ataques.

Inicialmente, se partió de la idea general de que la herramienta a desarrollar tendría como cometido la validación de la implementación de la comunicación DLMS/COSEM entre los medidores, concentradores y el Head-End System (HES), en términos de requisitos de seguridad. Surgieron así las siguientes interrogantes:

- ¿Es viable una herramienta de estas características?
- ¿Cómo se realizará la validación de la implementación de la comunicación DLMS/COSEM?
- ¿Cómo será la interacción entre la herramienta y los medidores o concentradores?
- ¿Cuál será la arquitectura de la solución y cuáles tecnologías serán necesarias?
- ¿Cuáles son los requerimientos de la herramienta?
- ¿Cómo se validará la herramienta al no contar con un medidor físico?

A continuación, se detalla el análisis realizado, donde se evaluaron las preguntas antes planteadas, se consideraron opciones, y se llegó a una visión de la herramienta más precisa. Por último, se presentan los requerimientos definidos para la herramienta, como resultado de esta etapa de análisis.

3.1. Análisis del problema

Para comenzar esta etapa se decidió estudiar herramientas similares, evaluando su implementación, con el objetivo de obtener ideas sobre cómo implementar una herramienta que cumpla con el objetivo planteado. En particular, se realizó un análisis y acercamiento al proyecto Metasploit [26], especialmente a su herramienta de código abierto llamada Metasploit Framework. La misma se trata de una aplicación desarrollada en Ruby que brinda funcionalidades para el desarrollo y ejecución de código exploit contra víctimas remotas. De esta manera, se analizó cómo estaba implementada la aplicación y se intentó tomar ideas que fuesen útiles para la herramienta. Entre algunas de las ideas que surgieron se encuentran:

- Desarrollar la herramienta utilizando Ruby como lenguaje, y así poder reutilizar partes del código de la aplicación. Uno de los desafíos a resolver en este caso hubiese sido cómo manipular la generación e interpretación de los mensajes. Fue clara entonces la necesidad de contar con un cliente DLMS/COSEM en Ruby, pero su búsqueda en la web no fue exitosa en su momento. Como alternativa, se evaluó la posibilidad de importar código en otro lenguaje, por ejemplo Java, en una aplicación Ruby.
- La aplicación está estructurada en base a módulos de distinta índole que ofrecen distintas funcionalidades útiles para la ejecución de los exploits. Se ofrece así la capacidad de extender las capacidades de la aplicación a través de módulos desarrollados por el usuario, la cual se consideró que podría ser una funcionalidad aplicable también a la herramienta a desarrollar.

En [27], los autores presentan el framework ValiDLMS para la validación de una implementación de DLMS/COSEM. En este se definen dos componentes principales, Fuzzer y Verifier, que se encargan de realizar ataques sobre el sistema y de verificar mensajes capturados en la red, respectivamente. El Fuzzer toma como entrada ataques y exploits almacenados en una base de datos de ataques, mientras que el Verifier cuenta con una base de datos de vulnerabilidades e información sobre la especificación de DLMS para poder identificar un paquete formado correctamente y

vulnerabilidades conocidas.

Se consideró entonces utilizar dicho framework. Nos comunicamos con el autor, quien nos envió una prueba de concepto solamente del Verifier, ya que no contaba con una implementación del Fuzzer. La misma se componía de un dissector para DLMS/COSEM sobre redes HDLC implementado como un plugin para Wireshark [28] desarrollado en Lua [29], que además señalaba mediante coloración en la interfaz de Wireshark posibles errores o warnings de seguridad en los mensajes. En un principio se creyó que podría ser de utilidad, ya que se contaba con un dissector de DLMS/COSEM para redes TCP y era posible integrarlos. Es así que se definió la siguiente arquitectura preliminar:

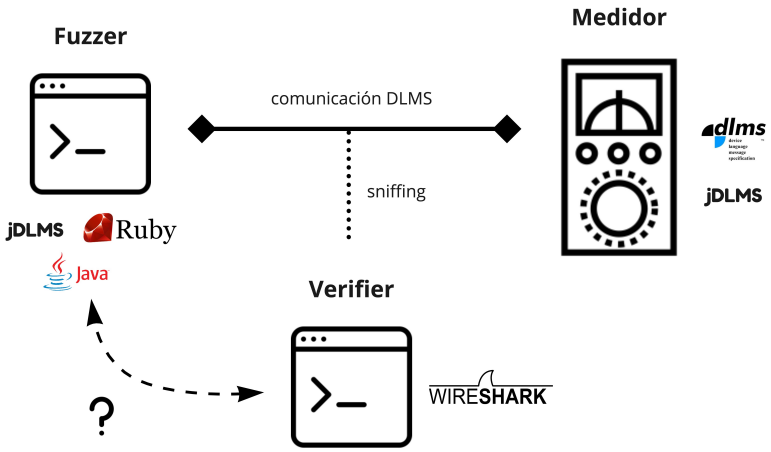

Figura 3.1: Arquitectura preliminar

En este diagrama se puede observar que la idea, siguiendo el framework ValiDLMS, era tener dos componentes individuales, Fuzzer y Verifier. El Fuzzer estaría encargado de enviarle mensajes al medidor para la ejecución de ataques. Para su implementación se consideraron dos opciones, implementarlo en Ruby y reutilizar código de Metasploit o implementarlo en Java y utilizar jDLMS [30], una librería Java que implementa el protocolo DLMS/COSEM. Por otro lado, el Verifier sería implementado utilizando los plugins para Wireshark mencionados anteriormente, y se encargaría de escuchar la comunicación entre Fuzzer y medidor y verificar si el ataque fue exitoso o si se encuentran posibles vulnerabilidades. Aún no resultaba claro cómo se llevaría a cabo la interacción entre ambos componentes.

Sin embargo, debido a que el análisis realizado por el plugin era solamente un análisis estático del tráfico y no proveía la capacidad de tomar acciones en base a dicho análisis, se consideró que no era una opción adecuada para nuestros objetivos.

Fue así que luego de un análisis más profundo se decidió que el framework no se ajustaba a los objetivos y necesidades. El framework en sí estaba descrito a alto nivel y hubiese sido necesario analizar todos los detalles de su implementación desde cero, debido a que la prueba de concepto enviada por el autor no fue de utilidad. Durante esta etapa de análisis, se consideró que era una mejor opción enfocarse en definir una implementación más concreta.

De todas formas, a partir del estudio de este framework, fue posible visualizar con más claridad cómo sería el funcionamiento de la herramienta a la hora de validar la implementación del protocolo. En forma similar a ValiDLMS, se decidió realizar la validación mediante la ejecución automatizada de ataques sobre medidores y/o concentradores, enviando mensajes DLMS/COSEM a los mismos y verificando las respuestas. Además, fue claro que debíamos definir una arquitectura y tecnologías que permitieran enviar mensajes al medidor, analizar las respuestas del mismo y en base a éstas decidir qué mensajes enviar a continuación.

Finalmente, se decidió que lo mejor sería implementar la herramienta en Java, de forma de poder contar con las funcionalidades ofrecidas por jDLMS. La opción de implementarla en Ruby resultaba interesante para poder reutilizar módulos de metasploit pero se decidió no utilizarlo debido a que en principio no se necesitaba ninguno de ellos en particular, y a que la búsqueda de un cliente DLMS en Ruby no fue exitosa.

La herramienta enviaría mensajes al medidor, recibiría las respuestas y en cada ataque se podría analizar las mismas y definir lógica en base a las respuestas recibidas. Se tendría así un único componente que integre ciertas funcionalidades del Fuzzer y del Verifier, sin ser independientes.

Como siguiente aspecto a resolver, fue necesario definir cómo iban a estar implementados los ataques o cómo los usuarios iban a ser capaces de ingresar o seleccionar el ataque a ejecutar. Para eso se consideraron dos opciones:

- Que los ataques fueran ingresados por el usuario en forma de árboles de ataque o alguna estructura similar. Esta idea surgió a partir del trabajo en el proyecto con UTE, donde se estaban definiendo árboles de ataque sobre la infraestructura del sistema de medidas como parte del análisis de seguridad. De seguir este camino, se consideró la necesidad de definir una estructura de datos general para los árboles de ataque, que permita el mapeo entre estos y acciones específicas a ejecutar, de forma de poder expresar en la herramienta un árbol en particular.

- Proveer al usuario ataques ya integrados en la herramienta y disponer de pasos de ataque que al ser combinados sirvan para definir ataques. A partir de esto, el usuario podría ejecutar uno de los ataques ya implementados o ejecutar los pasos de ataque en el orden deseado en forma manual.

Al considerar la primera opción, se intentó partir de uno de los árboles de ataque generados en el proyecto con UTE e implementarlo en la herramienta pero el mismo no contaba con el nivel de refinamiento necesario. Las hojas contenían acciones de un nivel de abstracción demasiado alto para lo que se necesitaba. De esta manera, se advirtió que resultaba complejo determinar pasos y acciones concretas que sean automatizables a partir de árboles de ataque con alto nivel de abstracción, como con el que se contaba. Surge así la necesidad de contar con pasos conocidos aún más pequeños que sean fáciles de implementar. Por lo tanto, se decidió seguir adelante con la segunda opción.

Para definir los pasos de ataque se decidió tomar como base el análisis de las vulnerabilidades conocidas. Para esto, fue necesario investigar sobre vulnerabilidades y ataques ya conocidos, tanto investigando en bases de datos públicas de vulnerabilidades como en la bibliografía sobre DLMS/COSEM. No se encontraron resultados relacionados a DLMS/COSEM en las bases de datos de vulnerabilidades. Además, en toda la bibliografía consultada, solamente en [25] se encontró un análisis de las vulnerabilidades, tanto del protocolo como de sus implementaciones. Esta falta de información nos induce a pensar que la temática de seguridad en smart meters aún no ha sido lo suficientemente estudiada y desarrollada, especialmente para el protocolo particular.

Como funcionalidad adicional, se decidió proveer al usuario la posibilidad de ejecutar pruebas para comprobar la presencia de las vulnerabilidades conocidas estudiadas. Además, tomando como inspiración el caso de metasploit, el usuario podría extender las capacidades de la herramienta implementando nuevos pasos de ataque, pruebas de vulnerabilidad o incluso nuevos ataques implementando nuevos módulos.

Como último punto a resolver, fue necesario contar con un servidor DLMS/COSEM que permitiera validar la implementación de la herramienta. Debido a que no se contaba con un medidor físico, se buscó implementar un servidor propio para prescindir del mismo.

Además, resultó clara la necesidad de implementar en el servidor las vulnerabilidades estudiadas, de forma de poder validar de mejor manera los ataques y pruebas de vulnerabilidad. Para esto, como parte de esta etapa y a modo de prueba de con-

cepto para evaluar la viabilidad de esta opción, se implementó el servidor en Java utilizando jDLMS y se modificó su funcionamiento para implementar algunas de las vulnerabilidades. Además, esto sirvió para probar enviar mensajes a este servido, analizar la comunicación entre ambos, y validar así la idea definida para la herramienta.

3.2. Requerimientos definidos

A partir del trabajo realizado en la etapa anterior, donde se analizaron distintas visiones de la herramienta y enfoques de implementación, se logró llegar a la siguiente idea para la misma.

El usuario de la herramienta, podrá conectarse a un servidor DLMS/COSEM, ejecutar varias pruebas de vulnerabilidades, comprobar cuáles se encuentran presentes, y en base a este análisis, podrá decidir ejecutar alguno de los ataques disponibles, crear sus propios ataques o ejecutar pasos de ataque en forma individual. Con el uso de esta herramienta el tiempo de pruebas se reducirá, además de resultar más sencillo, evitando el análisis del tráfico de datos, ya que cada prueba retornará el resultado al usuario, permitiéndole saber rápidamente cuáles de las pruebas ejecutadas fueron exitosas.

Se definieron así, en forma específica, las siguientes funcionalidades o requerimientos:

- Ingresar datos del medidor víctima
  El usuario es capaz de ingresar host y puerto del medidor que desea atacar.
- Listar pruebas de vulnerabilidades disponibles
  El usuario es capaz de listar las pruebas de vulnerabilidades que se encuentran disponibles para ejecutar.
- Ejecutar prueba de vulnerabilidad
  El usuario es capaz de seleccionar una prueba de vulnerabilidad y ejecutarla.
- Listar pasos de ataque
  El usuario es capaz de listar los pasos de ataque que se encuentran disponibles para ejecutar.
- Ejecutar paso de ataque
  El usuario es capaz de seleccionar un paso de ataque y ejecutarlo.

- Listar ataques disponibles
  El usuario es capaz de listar los ataques que se encuentran disponibles para ejecutar.

- Ejecutar ataque
  El usuario es capaz de seleccionar un ataque y ejecutarlo.

- Realizar diagnóstico completo
  El usuario es capaz de realizar un diagnóstico completo sobre el medidor. El mismo consiste en la ejecución de todas las pruebas de vulnerabilidades disponibles y la presentación de resultados.

- Mostrar ayuda
  El usuario es capaz de obtener ayuda sobre cómo utilizar las distintas funcionalidades.

- Proveer interfaz de usuario
  El usuario es capaz de interactuar con la herramienta a través de una interfaz que facilite el uso de la misma.

- Proveer script de compilación y ejecución
  El usuario es capaz de compilar y ejecutar la herramienta fácilmente mediante la ejecución de un script. La compilación es necesaria en caso de que haya nuevos ataques, pruebas de vulnerabilidad o pasos de ataque implementados por el usuario.

Capítulo 4

Diseño e implementación

En esta sección se presentan las decisiones tomadas durante la etapa de diseño e implementación de la herramienta, conjunto con los detalles de implementación de las pruebas de vulnerabilidad y los ataques.

4.1. Decisiones de diseño

- La interfaz de usuario es basada en texto con interacción basada en comandos. A futuro puede ser deseable cambiar a otro tipo de interfaz, pero se optó por ésta debido a su simplicidad de uso y a que es común en herramientas similares. Además resulta sencilla de implementar, lo cuál permitió poner el foco en el análisis del protocolo, de las pruebas y ataques, y de su implementación.
- Se buscó implementar la herramienta en forma modularizada, de forma de fomentar la reutilización y legibilidad del código. A su vez, se buscó facilitar la implementación de nuevos ataques y pruebas de vulnerabilidades, proveyendo al usuario "pasos de ataque" previamente desarrollados y la capacidad de utilizarlos fácilmente. El usuario es capaz de crear nuevos ataques, pruebas de vulnerabilidad y pasos de ataque mediante la definición de nuevos módulos, los cuales son automáticamente integrados y ejecutables desde la interfaz.
- Como se mencionó en el capítulo 3, se decidió implementar la herramienta en java utilizando jDLMS para aprovechar las facilidades que esta librería provee. Si bien se utilizan algunas funciones o módulos de jDLMS para facilitar el armado de los mensajes a enviar, éstos son modificados según sea necesario para probar las distintas vulnerabilidades. Se utiliza un cliente propio para

realizar la comunicación con el medidor.

- Para el desarrollo de la herramienta se utilizó Maven [31], para facilitar el manejo de dependencias y el proceso de build de la misma.

4.2. Implementación

En el diagrama de la figura 4.1 se presenta el diagrama de clases de la herramienta.

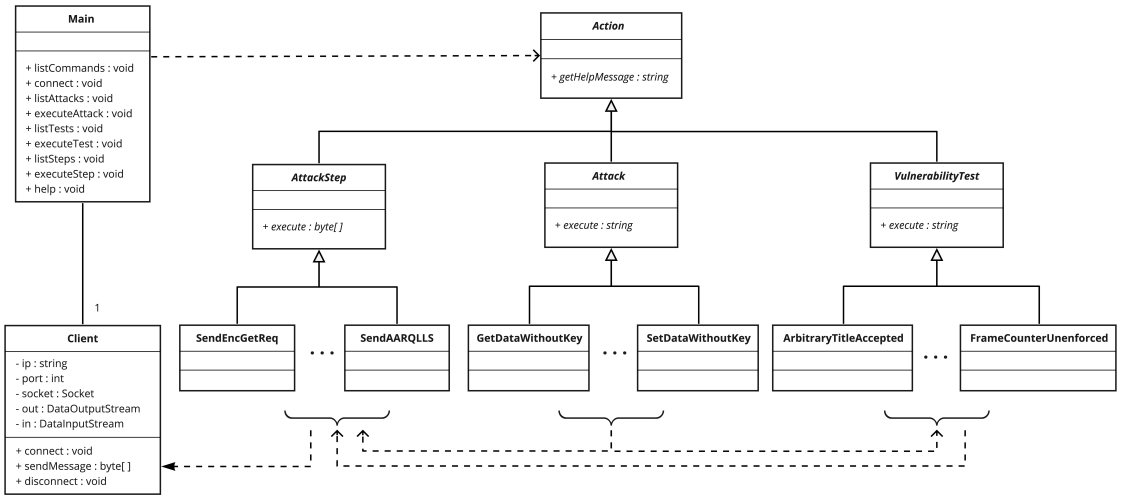

Figura 4.1: Diagrama de clases

Se tienen tres clases principales, cada una de ellas hereda de una clase padre `Action`, la cual representa un comando ejecutable a través de la interfaz:

- `AttackStep`
  Es una clase abstracta que representa pasos de ataque básicos que se utilizan para construir los ataques. La lógica del paso de ataque se definirá en el método abstracto `execute`, el cual retornará la respuesta del servidor como un arreglo de bytes. Para la comunicación con el servidor DLMS/COSEM, se utilizan los métodos de la clase `Client`.

- `Attack`
  Es una clase abstracta que representa un ataque ejecutable contra un servidor DLMS/COSEM. Tiene un método abstracto `execute` en el cual se definirá la lógica del ataque, el cual retornará la respuesta del servidor. Para esto se utilizarán los pasos de ataque descritos anteriormente, combinándolos en la forma deseada.

- `VulnerabilityTest`
  Se trata de una clase abstracta que representa una prueba de vulnerabilidad. Al igual que las dos clases anteriores cuenta con un método abstracto `execute` en el cual se encontrará la lógica de la prueba. Este método retornará un valor booleano indicando si la vulnerabilidad se encuentra presente o no.

A su vez, de cada una de estas clases se derivan clases particulares que implementan cada caso, permitiendo al usuario agregar nuevas pruebas de vulnerabilidad, pasos de ataque y ataques con solamente heredar de ellas e implementar el método `execute`. En el diagrama pueden verse algunas de las clases particulares implementadas.

Por último, la clase `Main` representa la interfaz de la aplicación donde se encuentran los distintos comandos ejecutables en la herramienta.

4.2.1. Pruebas de vulnerabilidades implementadas

Como se mencionó anteriormente, para implementar una prueba de vulnerabilidad se crea una nueva clase que derive de la clase abstracta `VulnerabilityTest`. A partir del estudio de la bibliografía consultada [25], se implementaron:

ArbitraryTitlesAccepted

Objetivo: Comprobar si un servidor acepta comunicaciones desde cualquier dispositivo. Es decir, desde dispositivos con cualquier AP Title.

Pasos a ejecutar:

1. Se envía un mensaje AARQ en texto plano con un AP Title generado en forma aleatoria.
2. En base al AARE recibido se analiza si la comunicación fue aceptada por el servidor.

FrameCounterUnenforced

Objetivo: Comprobar si un servidor acepta mensajes cuyo frame counter sea menor o igual al contador del último mensaje cifrado recibido de un dispositivo

Pasos a ejecutar:

1. Se envía un mensaje AARQ en texto plano para comenzar la comunicación con el medidor, el cual debido a las vulnerabilidades presentes responde con un mensaje AARE cifrado.
2. De este mensaje recibido, se obtiene el desafío HLS y el AP Title enviados por el servidor.
3. Siguiendo el funcionamiento del algoritmo de cifrado utilizado por DLMS/-COSEM explicado en la sección 3.5, se obtiene el keystream realizando un xor entre el payload del mensaje AARE cifrado recibido y el payload esperado. Esto es posible debido a que los mensajes AARE son predecibles en base al AARQ enviado.
4. Con la información obtenida se calcula la respuesta al desafío de autenticación enviado por el servidor, utilizando la clave de autenticación conocida previamente y el AP Title y el desafío recibidos desde el servidor.
5. Se envía un mensaje Action Request cifrado sin actualizar el valor del frame counter. Es posible cifrar el mensaje sin conocer la clave de cifrado debido al keystream obtenido en el paso 3.
6. Se verifica si la comunicación fue aceptada en base al mensaje Action Response recibido.

IdenticalAATitlesAllowed

Objetivo: Comprobar si se permiten comunicaciones con dos dispositivos con el mismo AP Title.

Pasos ejecutados:

1. Se envía un mensaje AARQ en texto plano.
2. Del mensaje AARE recibido se obtiene el AP Title del servidor.
3. Se cierra la conexión enviando un mensaje Release Request.
4. Se envía un nuevo mensaje AARQ en texto plano con el AP Title obtenido en el paso 2.
5. En base al mensaje AARE recibido se verifica si la comunicación fue aceptada.

MessageAuthenticationNotEnforced

Objetivo: Comprobar si se permiten mensajes con un tag de autenticación inválido.

Pasos a ejecutar:

1. Se envía un mensaje AARQ en texto plano para comenzar la comunicación con el servidor, el cual debido a las vulnerabilidades presentes responde con un mensaje AARE cifrado.
2. De este mensaje recibido, se obtiene el desafío HLS y el AP Title enviados por el medidor.
3. Siguiendo el funcionamiento del algoritmo de cifrado utilizado por DLMS/-COSEM explicado en la sección 3.5, se obtiene el keystream realizando un xor entre el payload del mensaje AARE cifrado recibido y el payload esperado. Esto es posible debido a que los mensajes AARE son predecibles en base al AARQ enviado.
4. Con la información obtenida se calcula la respuesta al desafío de autenticación enviado por el servidor, utilizando la clave de autenticación conocida previamente y el AP Title y el desafío recibidos desde el servidor.
5. Se envía un mensaje Action Request cifrado sin el campo de autenticación. Es posible cifrar el mensaje sin conocer la clave de cifrado debido al keystream obtenido en el paso 3.
6. Se verifica si la comunicación fue aceptada en base al mensaje Action Response recibido.

PlaintextAARQCipheredAAREPresent

Objetivo: Comprobar si el servidor envía mensajes AARE cifrados en respuesta a mensajes AARQ en texto plano.

Pasos a ejecutar:

1. Se envía un AARQ en texto plano.
2. Se verifica si el AARE recibido se encuentra cifrado o no.

PlaintextAuthAllowed

Objetivo: Comprobar si se permite el uso de LLS como método de autenticación.

Pasos a ejecutar:

1. Envía un paquete AARQ en texto plano, solicitando que la comunicación utilice autenticación en texto plano (Low Level Security).
2. En base al AARE recibido se verifica si la comunicación fue aceptada por el servidor.

4.2.2. Ataques implementados

Para implementar un ataque, se crea una nueva clase que derive de la clase abstracta Attack. Los ataques implementados son los siguientes:

GetDataWithoutKey

Objetivo: Obtener los datos de medida de energía sin contar con la clave de cifrado. Para que este ataque sea exitoso las siguientes vulnerabilidades deben estar presentes:

- El servidor permite la comunicación con dispositivos con su mismo AP Title.
- El servidor envía mensajes AARE cifrados en respuesta a AARQ en texto plano.
- El servidor permite mensajes cifrados sin autenticación.

Además, debe conocerse la clave de autenticación. Esto a priori puede parecer improbable pero en la realidad algunos medidores utilizan claves de autenticación por defecto.

Pasos a ejecutar:

1. Se envía un mensaje AARQ en texto plano para comenzar la comunicación con el medidor, el cual debido a las vulnerabilidades presentes responde con un mensaje AARE cifrado.
2. De este mensaje recibido, se obtiene el desafío HLS y el AP Title enviados por el servidor.
3. Siguiendo el funcionamiento del algoritmo de cifrado utilizado por DLMS/-COSEM explicado en la sección 3.5 se obtiene el keystream realizando un xor entre el payload del mensaje AARE cifrado recibido y el payload esperado. Esto es posible debido a que los mensajes AARE son predecibles en base al AARQ enviado.
4. Con la información obtenida se calcula la respuesta al desafío de autenticación enviado por el servidor, utilizando la clave de autenticación conocida previamente y el AP Title y el desafío recibidos desde el servidor.
5. Se envía un mensaje Action Request cifrado. Es posible cifrar el mensaje sin conocer la clave de cifrado debido al keystream obtenido en el paso 3. Luego de este paso se logra tener una comunicación autenticada.
6. Se envía un mensaje Get Request cifrado, también mediante el keystream

obtenido en el paso 3. Se reciben los datos de medida de energía.

7. Se envía un mensaje Release Request para finalizar la comunicación con el servidor.

SetDataWithoutKey

Este ataque sigue la misma lógica que el ataque anterior pero en lugar de obtener los datos de medida busca modificar los mismos. Las precondiciones necesarias para ejecutar este ataque son las mismas que para el ataque anterior. A su vez, los pasos a realizar para su ejecución también son los mismos con la excepción del paso 6, en el cual se envía un mensaje Set Request en lugar de un Get Request.

4.2.3. Pasos de ataque implementados

A la hora de implementar las pruebas y ataques descritos anteriormente, se implementaron los siguientes pasos de ataque, que derivan de la clase AttackStep:

- *SendAARQPlaintext*
  Envía un paquete AARQ en texto plano, opcionalmente se puede indicar el AP Title a enviar.
- *GetSTOC*
  A partir de un mensaje AARE recibido, retorna el desafío enviado desde el servidor (medidor) al cliente (Server to Challenge Response).
- *GetServerTitle*
  A partir de un mensaje AARE recibido, retorna el AP Title enviado por el servidor.
- *GetKeystream*
  A partir de un mensaje AARE cifrado recibido, calcula y retorna el keystream utilizado para cifrar el mensaje.
- *GetChallengeResponse*
  A partir de la clave de autenticación, el AP Title y el desafío recibidos del servidor, calcula y retorna la respuesta a dicho desafío.
- *SendEncActionRequest*
  Tomando como parámetros el keystream y la respuesta al desafío de autenticación, envía un mensaje Action Request cifrado.

- *SendEncGetRequest*
  Tomando como parámetros el keystream, envía un mensaje Get Request cifrado.
- *SendEncSetRequest*
  Tomando como parámetros el keystream, envía un mensaje Set Request cifrado.
- *SendRLRQ*
  Envía un mensaje Release Request.
- *RemoveAuthentication*
  Tomando un mensaje como parámetro, retorna el mismo mensaje sin el campo de autenticación.
- *SendAARQLLS*
  Envía un paquete AARQ en texto plano, solicitando que la comunicación utilice autenticación en texto plano (Low Level Security).

En el apéndice 2 se puede consultar el manual de usuario de la herramienta, donde se ofrece una descripción de cada uno de los comandos ofrecidos por la misma.

Capítulo 5

Verificación

En esta sección se presenta el trabajo realizado en la etapa de verificación de la herramienta. Como se mencionó anteriormente en el capítulo de análisis, debido a la falta de un medidor físico que permitiera verificar el funcionamiento de la misma, se decidió implementar un servidor DLMS/COSEM propio e introducir las vulnerabilidades necesarias para llevar a cabo las pruebas.

5.1. Implementación de servidor vulnerable

Con el fin de contar con un servidor contra el cual ejecutar la herramienta, y extendiendo la prueba de concepto realizada durante la etapa de análisis, se implementó utilizando jDLMS. Para satisfacer las necesidades particulares de esta etapa, es decir, probar las pruebas de vulnerabilidades y los ataques implementados en la herramienta, fue necesario realizar modificaciones al código de jDLMS de forma de introducir algunas de las vulnerabilidades estudiadas. Las vulnerabilidades que se buscó introducir fueron las siguientes:

- *Mensajes AARE cifrados en respuesta a mensajes AARQ en texto plano*
  Originalmente el comportamiento del servidor utilizando jDLMS era responder con un mensaje AARE en texto plano o cifrado según si el mensaje AARQ era en texto plano o cifrado, respectivamente. Para introducir la vulnerabilidad en el servidor, se modificó jDLMS para que siempre responda mensajes AARE cifrados, sin importar si el AARQ se encuentra cifrado o no.
- *Se permiten comunicaciones con dispositivos con el mismo AP Title que el servidor*

Fue necesario realizar modificaciones a jDLMS para introducir esta vulnerabilidad, originalmente las comunicaciones de este tipo eran rechazadas.

- *Se permiten mensajes cifrados sin autenticación*
  Se comprobó que jDLMS verifica la presencia y validez del tag de autenticación. Al enviar mensajes cifrados a los cuales se les removió dicho tag, los mismos fueron rechazados. Al investigar el código, se encontró que jDLMS utiliza una librería que provee las funciones criptográficas, mediante el uso de la cual no es posible descifrar un mensaje si no cuenta con el tag de autenticación. Se buscó el código fuente de la librería y se intentó modificar la función de descifrado para lograr el comportamiento deseado sin éxito debido a la complejidad de la implementación del algoritmo. Por lo tanto, fue necesario imitar el comportamiento deseado en jDLMS con mensajes pre armados, obviando el uso de esta librería para nuestros ataques particulares. Esto implica que no fue posible realizar una verificación completa de uno de los pasos de ataque implementados y por lo tanto del ataque en particular implementado, ya que dicho ataque requiere la presencia de esta vulnerabilidad. Quedará como trabajo a futuro lograr realizar los cambios en la librería que permitan verificar completamente la correctitud de este ataque.

- *Se permiten comunicaciones con dispositivos con AP Title arbitrarios. Es decir, las comunicaciones no se limitan a un conjunto de dispositivos permitidos*
  Fue necesario realizar modificaciones a jDLMS para introducir esta vulnerabilidad, originalmente las comunicaciones de este tipo eran rechazadas.

- *Se aceptan mensajes cuyo frame counter sea menor o igual al contador del último mensaje cifrado recibido de un dispositivo*
  El comportamiento esperado de DLMS es que se rechacen mensajes cuyo frame counter sea menor al frame counter del último mensaje recibido. Por lo tanto, fue necesario modificar la implementación de jDLMS para poder contar con esta vulnerabilidad.

- *Se aceptan mensajes con tag de autenticación inválido*
  Por los mismos motivos que se explicaron anteriormente para la vulnerabilidad "Se permiten mensajes cifrados sin autenticación", fue necesario imitar el comportamiento deseado en el servidor vulnerable.

- *Se permiten comunicaciones con métodos de autenticación inseguros*
  No fue necesario realizar modificaciones a jDLMS, se puede setear mediante un parámetro de configuración al instanciar el servidor.

Las primeras tres vulnerabilidades son precondición de los ataques implementados y el resto se implementaron para poder probar el resto de pruebas de vulnerabilidad.

5.2. Plan de pruebas

En esta sección se presenta el plan de pruebas ejecutado sobre el servidor vulnerable, de forma de poder verificar la correctitud de la herramienta.

Antes de ejecutar cualquiera de las pruebas es necesario haberse conectado previamente al servidor mediante el comando `connect` indicando host y puerto: `connect localhost 4059`.

Prueba 1

Ejecutar prueba de vulnerabilidad ArbitraryTitlesAccepted mediante el comando `execute-test ArbitraryTitlesAccepted`.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test ArbitraryTitlesAccepted
Vulnerability is present
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test ArbitraryTitlesAccepted
Vulnerability is not present
DLMSAnalyzer>
```

127.0.0.1	127.0.0.1	DLMS	350	nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1	127.0.0.1	DLMS	232	nBox DLMS 66 Bytes: AARE Logical_Name rejected-permanent

En las capturas de Wireshark se puede observar cómo la comunicación fue aceptada por el servidor vulnerable mientras que fue rechazada por el servidor no vulnerable. Por lo tanto, la herramienta detecta correctamente la presencia de la vulnerabilidad, retornando al usuario el resultado de la prueba en forma correcta.

Prueba 2

Ejecutar prueba de vulnerabilidad FrameCounterUnenforced mediante el comando `execute-test FrameCounterUnenforced`, donde nos será solicitado ingresar la clave de autenticación.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test FrameCounterUnenforced
Enter value of parameter authenticationKey. If left blank the test FrameCounterU
nenforced will not be executed:
PasswordPassword
Vulnerability is present
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	103	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	111	nBox DLMS 47 Bytes: glo-action-response(SS_0 AuthEncr Ucast FC=2)

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test FrameCounterUnenforced
Enter value of parameter authenticationKey. If left blank the test FrameCounterUnenforced will not be executed:
PasswordPassword
Vulnerability is not present
DLMSAnalyzer>
```

```
127.0.0.1    127.0.0.1    DLMS    388 nBox DLMS 144 Bytes: AARQ Logical_Name_With_Ciphering high_level_G
127.0.0.1    127.0.0.1    DLMS    396 nBox DLMS 148 Bytes: AARE Logical_Name_With_Ciphering accepted
127.0.0.1    127.0.0.1    DLMS    202 nBox DLMS  51 Bytes: glo-action-request(SS_0 AuthEncr Ucast FC=1)
```

Se observa que al enviar un segundo mensaje manteniendo el frame counter en 1, el mensaje es aceptado por el servidor vulnerable pero rechazado por el servidor no vulnerable, quien no envía respuesta. Por lo tanto, la herramienta detecta correctamente la presencia de la vulnerabilidad, retornando al usuario el resultado de la prueba en forma correcta.

Prueba 3

Ejecutar prueba de vulnerabilidad IdenticalAATitlesAllowed ejecutando el comando `execute-test IdenticalAATitlesAllowed`.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test IdenticalAATitlesAllowed
Vulnerability is present
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRQ
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRE
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRQ
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRE

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test IdenticalAATitlesAllowed
Vulnerability is not present
DLMSAnalyzer>
```

```
127.0.0.1       127.0.0.1       DLMS     350 nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1       127.0.0.1       DLMS     232 nBox DLMS  66 Bytes: AARE Logical_Name rejected-permanent
127.0.0.1       127.0.0.1       DLMS     110 nBox DLMS   5 Bytes: RLRQ
127.0.0.1       127.0.0.1       DLMS     350 nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1       127.0.0.1       DLMS     232 nBox DLMS  66 Bytes: AARE Logical_Name rejected-permanent
127.0.0.1       127.0.0.1       DLMS     110 nBox DLMS   5 Bytes: RLRQ
```

Como se observa en las capturas, primero se inicia una comunicación con el servidor para conocer su AP Title, tomándolo del mensaje AARE recibido. Esta primera comunicación es aceptada en el caso del servidor vulnerable, ya que se permiten comunicaciones con títulos arbitrario como se mostró en la prueba 1. En cambio, es rechazada en el caso del servidor no vulnerable porque éste no acepta comunicaciones con títulos arbitrarios. Luego, se inicia una nueva comunicación y la misma es aceptada por el servidor vulnerable y rechazada por el servidor no vulnerable, ya que la vulnerabilidad no se encuentra presente en éste último. Se concluye que la herramienta comprueba correctamente la presencia de la vulnerabilidad y retorna al usuario el resultado correcto.

Prueba 4

Ejecutar prueba de vulnerabilidad MessageAuthenticationNotEnforced ejecutando el comando `execute-test MessageAuthenticationNotEnforced`, donde nos será solicitado ingresar la clave de autenticación.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test MessageAuthenticationNotEnforced
Enter value of parameter authenticationKey. If left blank the test MessageAuthen
ticationNotEnforced will not be executed:
PasswordPassword
Vulnerability is present
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	103	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	111	nBox DLMS 47 Bytes: glo-action-response(SS_0 AuthEncr Ucast FC=2)

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test MessageAuthenticationNotEnforced
Enter value of parameter authenticationKey. If left blank the test MessageAuthenticationNotEnforced will not be executed:
PasswordPassword
Vulnerability is not present
DLMSAnalyzer>
```

127.0.0.1	127.0.0.1	DLMS	350	nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1	127.0.0.1	DLMS	232	nBox DLMS 66 Bytes: AARE Logical_Name rejected-permanent
127.0.0.1	127.0.0.1	DLMS	178	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)

Se puede observar que el servidor vulnerable acepta la conexión mientras que el servidor no vulnerable la rechaza. La herramienta devuelve el resultado de la prueba en forma correcta.

Prueba 5

Ejecutar prueba de vulnerabilidad PlaintextAARQCipheredAAREPresent ejecutando el comando `execute-test PlaintextAARQCipheredAAREPresent`.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test PlaintextAARQCipheredAAREPresent
Vulnerability is present
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted

```
  association_result  : accepted (0)
  association_diag    : null (0)
  responding-AP-title : 4d4d4d0000000001
  hls_StoC            : \037\357\277\275\\9*<1\037\357\277\275j\357\277\275\357\277\275\357\277\275\3
  user-information    : be2404222820300000000019f8cdf151b8823a1fc261e2eba...
> [Expert Info (Warning/Protocol): unknown user-information content: 0x22 Could be due to ciphering]
```

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test PlaintextAARQCipheredAAREPresent
Vulnerability is not present
DLMSAnalyzer>
```

127.0.0.1	127.0.0.1	DLMS	348	nBox DLMS 124 Bytes: AARQ Logical_Name high_level_GMAC
127.0.0.1	127.0.0.1	DLMS	354	nBox DLMS 127 Bytes: AARE Logical_Name accepted cnf_blk

```
    app_context         : Logical_Name (1)
    association_result  : accepted (0)
    association_diag    : null (0)
    responding-AP-title : 4d4d4d0000000001
    hls_StoC             [truncated]: \357\277\275\027tW\001\357
  > user-information    : be10040e0800065f1f040018123dffff0007
```

El resultado de la prueba fue el esperado, el servidor vulnerable responde con un AARE con texto cifrado, mientras que el servidor no vulnerable responde en texto plano. La herramienta detecta correctamente la presencia de la vulnerabilidad.

Prueba 6

Ejecutar prueba de vulnerabilidad PlaintextAuthAllowed ejecutando el comando `execute-test PlaintextAuthAllowed`.

Servidor vulnerable

Resultado esperado: El mensaje "Vulnerability is present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test PlaintextAuthAllowed
Vulnerability is present
DLMSAnalyzer>
```

```
127.0.0.1    127.0.0.1    DLMS    214 nBox DLMS  57 Bytes: AARQ Logical_Name low_level (Password)  cr
127.0.0.1    127.0.0.1    DLMS    194 nBox DLMS  47 Bytes: AARE Logical_Name_With_Ciphering accepted
```

Servidor no vulnerable

Resultado esperado: El mensaje "Vulnerability is not present" es desplegado al usuario.

Resultado obtenido:

```
DLMSAnalyzer> execute-test PlaintextAuthAllowed
Vulnerability is not present
DLMSAnalyzer>
```

```
127.0.0.1    127.0.0.1    DLMS    214 nBox DLMS  57 Bytes: AARQ Logical_Name low_level (Password)
127.0.0.1    127.0.0.1    DLMS    232 nBox DLMS  66 Bytes: AARE Logical_Name rejected-permanent
```

Como se puede observar en las capturas, en el caso del servidor vulnerable las comunicaciones con autenticación LLS son aceptadas mientras que el servidor no vulnerable no las acepta. La herramienta retorna correctamente al usuario el resultado de la prueba.

Prueba 7

Ejecutar ataque GetDataWithoutKey ejecutando el comando `execute-attack GetDataWithoutKey`, donde nos será solicitado ingresar la clave de autenticación.

Servidor vulnerable

Resultado esperado: El ataque es ejecutado con éxito contra el servidor vulnerable.

Resultado obtenido:

```
DLMSAnalyzer> execute-attack GetDataWithoutKey
Enter value of parameter authenticationKey. If left blank the test GetDataWithou
tKey will not be executed:
PasswordPassword
The response was:
0001000200010021cc1f30000000038bc2fcf87e83feb4678e3d2207cb314506d6958c9a48e8352d
4b
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	103	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	111	nBox DLMS 47 Bytes: glo-action-response(SS_0 AuthEncr Ucast FC=2)
127.0.0.1	127.0.0.1	DLMS	84	nBox DLMS 20 Bytes: glo-get-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	97	nBox DLMS 33 Bytes: glo-get-response(SS_0 AuthEncr Ucast FC=3)
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRQ
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRE

Servidor no vulnerable

Resultado esperado: El ataque no es exitoso.

Resultado obtenido:

```
DLMSAnalyzer> execute-attack GetDataWithoutKey
Enter value of parameter authenticationKey. If left blank the test GetDataWithoutKey will not be executed:
PasswordPassword
Connection was not accepted
Attack was not executed
DLMSAnalyzer>
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	350	nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1	127.0.0.1	DLMS	232	nBox DLMS 66 Bytes: AARE Logical_Name rejected-permanent
127.0.0.1	127.0.0.1	DLMS	178	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)

Prueba 8

Ejecutar ataque SetDataWithoutKey ejecutando el comando `execute-attack SetDataWithoutKey`, donde será solicitado ingresar la clave de autenticación y el nuevo valor a asignar como medida de energía.

Servidor vulnerable

Resultado esperado: El ataque es ejecutado con éxito contra el servidor vulnerable.

Resultado obtenido:

```
DLMSAnalyzer> execute-attack SetDataWithoutKey
Enter value of parameter authenticationKey. If left blank the test SetDataWithou
tKey will not be executed:
PasswordPassword
Enter value of parameter value. If left blank the test SetDataWithoutKey will no
t be executed:
12345
The response was:
0001000200010017cd1530000000038ac2fcf8c9411ff76b71995597485857
```

Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DLMS	189	nBox DLMS 125 Bytes: AARQ Logical_Name
127.0.0.1	127.0.0.1	DLMS	212	nBox DLMS 148 Bytes: AARE Logical_Name accepted
127.0.0.1	127.0.0.1	DLMS	103	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	111	nBox DLMS 47 Bytes: glo-action-response(SS_0 AuthEncr Ucast FC=2)
127.0.0.1	127.0.0.1	DLMS	93	nBox DLMS 29 Bytes: glo-set-request(SS_0 Encr Ucast FC=1)
127.0.0.1	127.0.0.1	DLMS	87	nBox DLMS 23 Bytes: glo-set-response(SS_0 AuthEncr Ucast FC=3)
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRQ
127.0.0.1	127.0.0.1	DLMS	69	nBox DLMS 5 Bytes: RLRE

Servidor no vulnerable

Resultado esperado: El ataque no es exitoso.

Resultado obtenido:

```
DLMSAnalyzer> execute-attack SetDataWithoutKey
Enter value of parameter authenticationKey. If left blank the test SetDataWithoutKey will not be executed:
PasswordPassword
Enter value of parameter value. If left blank the test SetDataWithoutKey will not be executed:
12345
Connection was not accepted
Attack was not executed
DLMSAnalyzer>
```

127.0.0.1	127.0.0.1	DLMS	350	nBox DLMS 125 Bytes: AARQ Logical_Name high_level_GMAC (��<N)
127.0.0.1	127.0.0.1	DLMS	232	nBox DLMS 66 Bytes: AARE Logical_Name rejected-permanent
127.0.0.1	127.0.0.1	DLMS	178	nBox DLMS 39 Bytes: glo-action-request(SS_0 Encr Ucast FC=1)

En las capturas de Wireshark para las dos pruebas anteriores se puede observar que la comunicación con el servidor vulnerable fue aceptada, por lo que los ataques se ejecutaron en forma exitosa. En cambio, en el caso del servidor no vulnerable, los ataques no pudieron ser ejecutados debido a la no presencia de las vulnerabilidades requeridas. Sin embargo, como se explicó anteriormente, una de las vulnerabilidades precondición de estos ataques no pudo ser implementada en el servidor por lo que fue necesario imitar el comportamiento deseado en jDLMS. Esto implica que no fue posible realizar una verificación completa de los ataques implementados, ya que uno de los pasos de estos ataques no pudo ser verificado correctamente.

Prueba 9

Ejecutar diagnóstico completo del medidor ejecutando el comando `vulnerability-check`.

Servidor vulnerable

Resultado esperado: Todas las pruebas son ejecutadas, para las pruebas que requieren parámetros los mismos son solicitados al usuario. 5 de las 6 vulnerabilidades se encuentran presentes. Una de ellas no se encuentra presente debido a que en esta oportunidad el servidor no estaba configurado para aceptar comunicaciones con métodos de autenticación inseguros.

Resultado obtenido:

```
DLMSAnalyzer> vulnerability-check
Enter value of parameter authenticationKey. If left blank the test MessageAuthenticationNotEnforced will not be executed:
PasswordPassword
MessageAuthenticationNotEnforced: Present
ArbitraryTitlesAccepted: Present
IdenticalAATitlesAllowed: Present
PlaintextAARQCipheredAAREPresent: Present
PlaintextAuthAllowed: Not Present
Enter value of parameter authenticationKey. If left blank the test FrameCounterUnenforced will not be executed:
PasswordPassword
FrameCounterUnenforced: Present

6 of 6 vulnerability test executed
5 of 6 vulnerabilities tested are present
DLMSAnalyzer>
```

Servidor no vulnerable

Resultado esperado: Todas las pruebas son ejecutadas. Para las pruebas que requieren parámetros los mismos son solicitados al usuario. Ninguna de las 6 vulnerabilidades se encuentra presente.

Resultado obtenido:

```
DLMSAnalyzer> vulnerability-check
Enter value of parameter authenticationKey. If left blank the test MessageAuthenticationNotEnforced will not be executed:
PasswordPassword
MessageAuthenticationNotEnforced: Not Present
ArbitraryTitlesAccepted: Not Present
IdenticalAATitlesAllowed: Not Present
PlaintextAARQCipheredAAREPresent: Not Present
PlaintextAuthAllowed: Not Present
Enter value of parameter authenticationKey. If left blank the test FrameCounterUnenforced will not be executed:
PasswordPassword
FrameCounterUnenforced: Not Present

6 of 6 vulnerability test executed
0 of 6 vulnerabilities tested are present
DLMSAnalyzer>
```

Prueba 10

Agregar una nueva prueba de vulnerabilidad.

Pasos:

1. Implementar una nueva prueba de vulnerabilidad extendiendo la clase `VulnerabilityTest`, implementando su método `execute`.

```
public class NewTest extends VulnerabilityTest {

  private String newAttribute;

  @Override
  public boolean execute(Client client) throws Exception {
    return !newAttribute.isEmpty();
  }

  @Override
  public String getHelpMessage() {
    return null;
  }

}
```

2. Listar los tests mediante el comando `list-tests`.
3. Ejecutar la nueva prueba mediante el comando *execute-test*.

Resultado esperado: La prueba se lista al ejecutar `list-tests` y es ejecutable mediante `execute-test`. De definirse atributos en la clase, los mismos serán solicitados al usuario para su ingreso.

Resultado obtenido:

```
DLMSAnalyzer> list-tests
Tests:
        MessageAuthenticationNotEnforced
        NewTest
        ArbitraryTitlesAccepted
        IdenticalAATitlesAllowed
        PlaintextAARQCipheredAAREPresent
        PlaintextAuthAllowed
        FrameCounterUnenforced
DLMSAnalyzer> execute-test NewTest
Enter value of parameter newAttribute. If left blank the test NewTest will not be executed:
value
Vulnerability is present
DLMSAnalyzer>
```

Prueba 11

Agregar un nuevo ataque.

Pasos:

1. Implementar un nuevo ataque extendiendo la clase `Attack`, implementando su método `execute`.

```
public class NewAttack extends Attack {

  private String newAttribute;

  @Override
  public String execute(Client client) throws Exception {
    return newAttribute;
  }

  @Override
  public String getHelpMessage() {
    return null;
  }

}
```

2. Listar los ataques mediante el comando `list-attacks`.
3. Ejecutar el nuevo ataque mediante el comando `execute-attack`.

Resultado esperado: El ataque se lista al ejecutar `list-attacks` y es ejecutable mediante `execute-attack`. De definirse atributos en la clase, los mismos serán solicitados al usuario para su ingreso.

Resultado obtenido:

```
DLMSAnalyzer> list-attacks
Attacks:
        SetDataWithoutKey
        NewAttack
        GetDataWithoutKey
DLMSAnalyzer> execute-attack NewAttack
Enter value of parameter newAttribute. If left blank the test NewAttack will not be executed:
value
The response was:
value
DLMSAnalyzer>
```

Prueba 12

Agregar un nuevo paso de ataque

Pasos:

1. Implementar un nuevo paso de ataque extendiendo la clase `AttackStep`, implementando su método `execute`.

```
public class NewAttackStep extends AttackStep {

  @Override
  public byte[] execute(Client client, Object... params)
  throws Exception {
    return new byte[3];
  }

  @Override
  public String getHelpMessage() {
    return null;
  }

}
```

2. Listar los pasos de ataque mediante el comando `list-steps`.
3. Ejecutar el nuevo paso de ataque mediante el comando `execute-step`.

Servidor vulnerable

Resultado esperado: El paso de ataque se lista al ejecutar `list-steps` y es ejecutable mediante `execute-step`.

Resultado obtenido:

```
DLMSAnalyzer> list-steps
Steps:
        SendEncActionReq
        SendAARQLLS
        SendAARQPlaintext
        GetKeystream
        SendEncGetReq
        GetServerTitle
        NewAttackStep
        SendRLRQ
        SendEncSetReq
        GetChallengeResponse
        RemoveAuthentication
        GetSTOC
DLMSAnalyzer> execute-step NewAttackStep
000000
DLMSAnalyzer>
```

Capítulo 6

Gestión y etapas del proyecto

En esta sección se describen las distintas etapas por las que transcurrió el proyecto, el tiempo dedicado a cada una y algunas decisiones generales tomadas en cada etapa. En un principio el proyecto estaba pensado para realizarse en un año pero, por diversos motivos personales y restricciones del proyecto, no se pudo cumplir lo estipulado y finalmente el tiempo total de duración fue de casi dos años.

En marzo de 2018 se comienza a evaluar conjunto con los tutores cuál será la propuesta de proyecto y su alcance. Se define así la colaboración en el proyecto conjunto con UTE, con planteos de objetivos iniciales pero no definitivos que serían especificados en detalle luego de conocer más de la temática y haber evaluado sobre qué enfocar el trabajo en el contexto del proyecto.

En abril se da comienzo formal al proyecto. En dicha primera etapa se estudió la bibliografía recabada sobre la temática, se participó de actividades conjunto con UTE y se evaluaron los distintos caminos posibles en los cuales seguir trabajando. Esta etapa de investigación llevó más tiempo del deseado debido a otros cursos realizados al mismo tiempo y a la dificultad encontrada a la hora de definir objetivos más específicos para el proyecto. En diciembre se logró definirlos y se comenzó así la siguiente etapa de análisis de la herramienta a desarrollar. En abril de 2019 se pone en pausa dicho trabajo, y surge la necesidad de trabajar en el análisis de seguridad de la solución Smart Grid de UTE. En agosto se da por terminada dicha etapa y se retoma el trabajo relacionado a la herramienta en sí, al mismo tiempo que se comienza la preparación del informe. En noviembre se da por terminado el desarrollo y verificación de la herramienta y se continúa con el informe. La preparación del informe llevó más tiempo del deseado y estimado debido a viajes de trabajo y, sobretodo, a la cantidad de información que se quiso desarrollar como estado del

arte, ya que uno de los objetivos del proyecto fue el relevamiento de IoT, Smart Grids y sus problemáticas de seguridad.

En la figura 6.1 se muestran las etapas y la duración de cada una, y en las siguientes secciones se detalla el trabajo realizado en ellas.

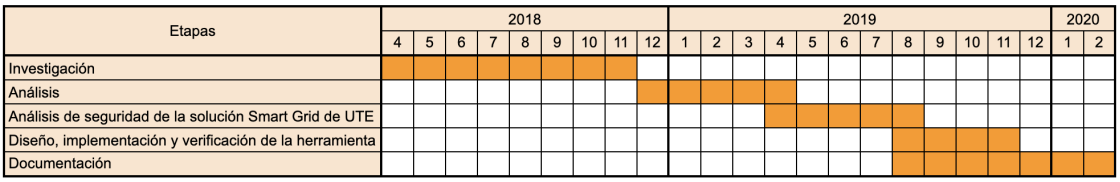

Figura 6.1: Etapas del proyecto

6.1. Investigación

Esta primera etapa consistió en el estudio de la bibliografía disponible para ahondar en el tema.

Uno de los primeros desafíos enfrentados fue no contar con conocimientos suficientes sobre IoT y sus problemáticas de seguridad, en especial para la realidad particular de las Smart Grid. Además, se trata de un área muy amplia y difícil de abarcar. Sumado a esto, el desarrollo de conocimiento en el área es reciente en comparación con otras áreas de las tecnologías de la información. Es así que se encontró que la bibliografía disponible sobre estos temas es escasa y muy heterogénea en enfoques y en contenido.

Dentro de esta etapa, como parte de las actividades enmarcadas en el proyecto con UTE, se participó de seminarios conjunto con empleados de la empresa sobre distintos temas referentes a IoT y Smart Grid, como forma de generar un espacio de aprendizaje e intercambio de conocimiento entre los participantes.

Luego de este estudio inicial del área, se prosiguió con el estudio del protocolo DLMS/COSEM, haciendo énfasis en sus problemáticas de seguridad, estudiando vulnerabilidades del protocolo en sí y de sus implementaciones.

A partir del estudio del protocolo, se comenzó a evaluar qué líneas de trabajo se podrían seguir para trabajar en la detección y explotación de sus vulnerabilidades, y en particular las posibilidades de automatización de estas actividades.

Como fin de esta etapa, se propuso el desarrollo de la herramienta que brinda

soporte automatizado a la detección de vulnerabilidades específicas al protocolo DLMS/COSEM.

6.2. Análisis del problema

Como punto de partida se definió que la herramienta a desarrollar tendría como cometido principal la validación de la implementación de la comunicación entre medidores, concentradores y el sistema de gestión, en términos de detección de vulnerabilidades.

En esta etapa fue necesario evaluar varios aspectos referentes a los requerimientos de la herramienta, la arquitectura de la solución, y herramientas y tecnologías a utilizar. Además, se estudiaron algunos frameworks propuestos en la bibliografía.

6.3. Análisis de seguridad de la solución Smart Grid de UTE

En el contexto del proyecto con UTE, se colaboró en la etapa de ejecución de pruebas sobre su solución de Smart Grid. Este trabajo implicó el entendimiento de su infraestructura y la evaluación sobre qué pruebas podrían realizarse y cómo. Esta etapa generó una pausa en el trabajo relacionado a la herramienta debido al modo de trabajo y al cronograma esperado. Luego de terminada, se continuó avanzando como es explicado en las siguientes secciones.

Dado que el enfoque de trabajo fue el protocolo DLMS/COSEM, se diseñaron y ejecutaron pruebas específicas sobre las comunicaciones entre los componentes de la solución. Además, se colaboró en el análisis de tráfico capturado en la infraestructura, lo que ayudó en el desarrollo de otras pruebas realizadas en el contexto del proyecto.

Los detalles de lo realizado en esta etapa y los resultados obtenidos se encuentran en el informe adjunto y no se detallan en el presente por motivos de confidencialidad.

6.4. Diseño, implementación y verificación de la herramienta

Luego de terminadas las pruebas en UTE se continuó con el trabajo realizado en la etapa de análisis, realizando el diseño e implementación de la herramienta. A su vez, se verificó el correcto funcionamiento de la misma contra el servidor vulnerable implementado.

Esta etapa tuvo una duración de cuatro meses, durante los cuáles gran parte del tiempo fue dedicado al estudio en detalle de los mensajes intercambiados en comunicaciones DLMS/COSEM. Fue necesario realizar pruebas para comprobar qué información era necesaria modificar, con el objetivo de atacar el medidor y comprobar la presencia de vulnerabilidades.

6.5. Documentación

Durante el transcurso del proyecto se fue dejando registro de las decisiones tomadas y de los resultados obtenidos. El informe general se comenzó a escribir en conjunto con la etapa de diseño e implementación de la herramienta, en el cual se describe en detalle cada etapa y sus resultados.

Para generar el estado del arte se utilizaron las distintas bibliografías consultadas a lo largo del proyecto, intentando consolidar los conocimientos adquiridos sobre la problemática de IoT y su seguridad, con un enfoque en el protocolo DLMS/COSEM que fue el objeto de estudio.

Capítulo 7

Conclusiones y trabajo a futuro

Al comienzo del proyecto se habían definido los siguientes objetivos:

- Realizar un relevamiento de IoT, Smart Grids y el protocolo DLMS/COSEM mediante el estudio de la bibliografía disponible.
- Relevar las problemáticas de seguridad de dicho protocolo.
- Contribuir a la detección de vulnerabilidades del protocolo que puedan ser utilizadas en la realización de ataques.
- Colaborar en el cumplimiento de los objetivos del proyecto con UTE.
- Desarrollar una herramienta que brinde soporte automatizado al proceso de detección de vulnerabilidades en el sistema de medidas.

Fue así que como punto de partida del proyecto se comenzó con el relevamiento de la temática. Se consultaron múltiples fuentes en búsqueda de lograr una mayor comprensión del área y de su problemática de seguridad. Además, en el contexto del proyecto con UTE, se participó de seminarios con expositores tanto de UTE como del resto del equipo de investigadores que también contribuyeron a este fin. En la sección de estado del arte y en el informe adjunto se presentaron los resultados de esta etapa, los cuales se consideran de gran utilidad para futuros trabajos en el área.

En cuanto al estudio de la seguridad del protocolo DLMS/COSEM, fue posible encontrar en la bibliografía consultada varias vulnerabilidades conocidas, las cuales fueron el insumo principal para el desarrollo de la herramienta. De todas formas, la bibliografía disponible no es demasiado extensa en cuanto a la seguridad del protocolo y Smart meters en general, lo cual resultó una dificultad en el proceso de desarrollo de la herramienta y en la colaboración en el proyecto con UTE.

Se colaboró junto al resto del equipo de investigadores en la realización de pruebas

en la infraestructura de Smart Grid de UTE, de forma de analizar la seguridad de su sistema de recolección de medidas. En particular, a partir del conocimiento adquirido sobre el protocolo DLMS/COSEM, se definieron distintos tipos de pruebas a ejecutar sobre la infraestructura. Se considera que los resultados obtenidos a partir de las mismas fueron de valor para UTE, así como para los autores del presente trabajo al permitir experimentar con el protocolo en una infraestructura real.

Durante el desarrollo de la herramienta se profundizó en el estudio del protocolo DLMS/COSEM, sobre todo en el análisis de la estructura de los mensajes y en el estudio en detalle de cada una de las vulnerabilidades y ataques. Si bien al principio fue complejo visualizar una solución que cumpliera los requerimientos que se habían definido, debido a su complejidad y alto nivel de abstracción, fue posible llegar a implementar un prototipo inicial de la herramienta.

El prototipo desarrollado permite la ejecución en forma automatizada de varias pruebas de vulnerabilidades conocidas y algunos ataques completos. De esta manera, permite evaluar las características de seguridad de un medidor en forma más rápida y sencilla, obteniendo resultados claros que no dependen del análisis del tráfico en forma manual. Además, se podrá continuar trabajando en el mismo para extender sus funcionalidades.

De acuerdo a los resultados presentados en el presente informe, es posible afirmar que se logró cumplir los objetivos definidos. Se adquirieron así nuevos conocimientos en el área, que fueron aplicados en el análisis de la seguridad de un sistema real de alta complejidad y en el desarrollo de un prototipo de herramienta que ayude a automatizar dicho proceso de análisis.

A continuación, se detallan las principales líneas de trabajo futuro que surgen a partir del trabajo realizado en el marco de este proyecto:

- Extender las funcionalidades de la herramienta
  - Implementar nuevas pruebas de vulnerabilidades y ataques.
  - Mejorar la funcionalidad de ejecución de pasos de ataque, permitiendo al usuario la ejecución de múltiples pasos en el orden que desee. De esta manera, no sería necesario la implementación de una clase nueva para implementar un ataque, sino que el usuario podría ejecutar ataques en forma dinámica, simplemente definiendo la lista de pasos a ejecutar. Para esto, las salidas y entradas de los `AttackStep` deberían estandarizarse, de forma de poder ser ejecutados en cualquier orden.
  - Modelar e implementar los ataques con una estructura de árbol, de forma

de lograr acercarse a cumplir el requerimiento inicial de que la entrada de la herramienta fuese un árbol de ataque a ejecutar. Actualmente los ataques utilizan distintos pasos de ataque, que el usuario eventualmente puede dar un comportamiento de árbol utilizando lógica condicional, pero que no se encuentran modelados de esa manera directamente. Sería deseable entonces que se pueda definir un árbol de `AttackSteps` y que la herramienta lo tome como entrada.

- Utilizar la herramienta en un sistema de recolección de medidas de energía, de forma de poder validarla en un contexto real. De ser posible, sería deseable ejecutarla en la infraestructura de UTE.
- Continuar profundizando el conocimiento sobre el protocolo y su seguridad. Si bien se trabajó con vulnerabilidades encontradas en la bibliografía, no fue posible descubrir nuevos tipos de vulnerabilidades en base la definición o implementaciones del protocolo.

Referencias bibliográficas

[1] Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, and Rob Kranenburg. Enabling things to talk: designing IoT solutions with the IoT architectural reference model. Springer, 2013.

[2] European Union Agency for Network and Information Security (ENISA). Baseline security recommendations for iot in the context of critical information infrastructures. Technical report, ENISA, Dec 2017.

[3] ISO/IEC JTC 1. Internet of things (iot) preliminary report 2014. Technical report, ISO/IEC JTC 1, 2014.

[4] George Corser. Internet of things (iot) security best practices. Technical report, Institute of Electrical and Electronics Engineers (IEEE), Feb 2017.

[5] Ericsson. Ericsson mobility report june 2019. Technical report, Ericsson, June 2019.

[6] International Electrotechnical Commission (IEC). IoT 2020: smart and secure IoT platform: white paper. International Electrotechnical Commission (IEC), 2016.

[7] Se-Ra Oh and Young-Gab Kim. Security requirements analysis for the iot. In 2017 International Conference on Platform Technology and Service (PlatCon), Feb 2017.

[8] Zozo Hassan, Hesham Ali, and Mahmoud Badawy. Internet of things (iot): Definitions, challenges, and recent research directions. International Journal of Computer Applications, 128:975–8887, Oct 2015.

[9] International Telecommunication Union (ITU). Global information infrastructure, internet protocol aspects and next-generation networks: Overview of the internet of things. Technical report, International Telecommunication Union (ITU), June 2012.

[10] Ravi C Bhaddurgatte and Vijaya Kumar BP. A review: Qos architecture and implementations in iot environment. In Trends in engineering and technological development: Some recent advances. Nov 2015.

[11] Industrial Internet Consortium. Industrial internet of things volume g4: Security framework. Technical report, Sep 2016.

[12] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. Computer Networks, 76:146 – 164, 2015.

[13] European Union Agency for Network and Information Security (ENISA). Enisa threat landscape report 2016. Technical report, ENISA, Jan 2017.

[14] European Regulators' Group for Electricity and Gas (ERGEG). Position paper on smart grids. an ergeg public consultation paper. Technical report, ERGEG, Dec 2009.

[15] National Institute of Standards and Technolgy (NIST). Framework and roadmap for smart grid interoperability standards, release 1.0. Technical report, NIST, 2009.

[16] Electric Power Research Institut (EPRI). Estimating the costs and benefits of the smart grids. a preliminary estimate of the investment requirements and the resulting benefits of a fully functioning smart grid. (technical report no. 1022519). Technical report, EPRI, 2011.

[17] Florian Skopik and Paul Smith. Smart grid security innovative solutions for a modernized grid. Elsevier, 2015.

[18] Unai Castro Legarza and Eloy Álvarez Pelegry. Redes de distribución eléctrica del futuro: Un análisis para su desarrollo. In Cuadernos Orkestra. Nov 2013.

[19] Smart Grid Coordination Group. Cencenelecetsi. reports in response to smart grid mandate m/490. Technical report, Smart Grid Coordination Group, 2014.

[20] Z. Popovic and V. Cackovic. Advanced metering infrastructure in the context of smart grids. Technical report, IEEE International Energy Conference (ENERGYCON), 2014.

[21] Christoph Ruland Obaid Ur-Rehman, Natasa Zivic. Security issues in smart metering systems. Technical report, Aug 2015.

[22] Mariagrazia Fugini Gerardo Pelosi Alessandro Barenghi, Luca Breveglieri. Computer Security Anchors in Smart Grids: The Smart Metering Scenario and Challenges. Springer, 2015.

[23] Kevin Fu Emmanuel Cecchet David Irwin Andrés Molina-Markham, Prashant Shenoy. Private memoirs of a smart meter. Technical report, 2010.

[24] Petr Matousek. Analysis of dlsm protocol. Technical report, Brno University of Technology, 2018.

[25] Loren Weith. Dlms / cosem protocol security evaluation. Thesis, Eindhoven University of Technology, March 2014.

[26] https://www.metasploit.com/.

[27] Henrique Califórnia Mendes. Security auditing of a dlms/cosem smart grid: Communication protocol implementation. Master's thesis, Faculdade de ciências, Universidade de Lisboa, 2018.

[28] https://www.wireshark.org/.

[29] https://www.lua.org/.

[30] https://www.openmuc.org/dlms-cosem/.

[31] https://maven.apache.org/index.html.

APÉNDICES

Apéndice 1

Ejemplos de mensajes DLMS/COSEM

A continuación se muestran ejemplos de la estructura de algunos mensajes DLMS/COSEM. En las siguientes figuras se podrá observar los campos que componen los mensajes, indicando cuáles son opcionales.

```
AARQ-apdu ::= [APPLICATION 0] IMPLICIT SEQUENCE {          --- APPLICATION 0 = 60H = 96
        protocol-version              [0]   IMPLICIT BITSTRING {version 1(0)} DEFAULT {version 1},
        application-context-name      [1]   Application-context-name,
        called-AP-title               [2]   AP-title OPTIONAL,
        called-AE-qualifier           [3]   AE-qualifier OPTIONAL,
        called-AP-invocation-id       [4]   AP-invocation-identifier OPTIONAL,
        called-AE-invocation-ide      [5]   AE-invocation-identifier OPTIONAL,
        calling-AP-title              [6]   AP-title OPTIONAL,
        calling-AE-quantifier         [7]   AE-qualifier OPTIONAL,
        calling-AP-invocation-id      [8]   AP-invocation-identifier OPTIONAL,
        calling-AE-invocation-id      [9]   AE-invocation-identifier OPTIONAL,
        sender-acse-requirements      [10]  IMPLICIT ACSE-requirements OPTIONAL,
        mechanism-name                [11]  IMPLICIT Mechanism-name OPTIONAL,
        calling-authentication-value  [12]  EXPLICIT Authentication-value OPTIONAL,
        implementation-information    [29]  IMPLICIT  Implementation-data OPTIONAL,
        user-information              [30]  IMPLICIT  Association-information OPTIONAL
}
```

Figura 1.1: Estructura mensaje AARQ

```
AARE-apdu ::= [APPLICATION 1] IMPLICIT SEQUENCE {          --- APPLICATION 1 = 61H = 97
        protocol-version                [0]     IMPLICIT BIT STRING {version 1(0)} DEFAULT {version1},
        application-context-name        [1]     Application-context-name,
        result                          [2]     Association-result,
        result-source-diagnostic        [3]     Associate-source-diagnostic,
        responding-AP-title             [4]     AP-title OPTIONAL,
        responding-AE-qualifier         [5]     AE-qualifier OPTIONAL,
        responding-AP-invocation-id     [6]     AP-invocation-identifier OPTIONAL,
        responding-AE-invocation-id     [7]     AE-invocation-identifier OPTIONAL,
        responder-acse-requirements     [8]     IMPLICIT ACSE-requirement OPTIONAL,
        mechanism-name                  [9]     IMPLICIT Mechanism-name OPTIONAL,
        responding-authentication-value [10]    EXPLICIT Authentication-value OPTIONAL,
        application-context-name-list   [11]    IMPLICIT Application-context-name-list OPTIONAL,
        implementation-information      [29]    IMPLICIT Implementation-data OPTIONAL,
        user-information                [30]    IMPLICIT Association-information OPTIONAL
}
```

Figura 1.2: Estructura mensaje AARE

```
RLRQ-apdu ::= [APPLICATION 2] IMPLICIT SEQUENCE {    --- APPLICATION 2 = 62H = 98
        reason                  [0]     IMPLICIT Release-request-reason OPTIONAL,
        aso-qualifier           [13]    ASO-qualifier OPTIONAL,
        asoi-identifier         [14]    IMPLICIT ASOI-identifier OPTIONAL,
        user-information        [30]    IMPLICIT Association-information OPTIONAL
}
```

Figura 1.3: Estructura mensaje RLRQ

```
RLRE-apdu ::= [APPLICATION 3] IMPLICIT SEQUENCE {     --- APPLICATION 3 = 63H = 99
        reason                  [0]     IMPLICIT Release=response-reason OPTIONAL,
        aso-qualifier           [13]    ASO-qualifier OPTIONAL,
        asoi-identifier         [14]    IMPLICIT ASOI-identifier OPTIONAL,
        user-information        [30]    IMPLICIT Association-information OPTIONAL
}
```

Figura 1.4: Estructura mensaje RLRE

Apéndice 2

Manual de usuario de la herramienta

El código de la herramienta puede encontrarse en: https://gitlab.fing.edu.uy/joaquin.marquez/dlmsanalyzer

2.1. Instrucciones de instalación

1. Clonar el proyecto de https://gitlab.fing.edu.uy/joaquin.marquez/dlmsanalyzer
2. Instalar Maven [31] y Java.
3. Ejecutar el script `dlms_analyzer.sh` para compilar y ejecutar la herramienta.

2.2. Comandos disponibles

A continuación se explican brevemente los comandos que ofrece la herramienta:

- `?list`
  Lista los comandos disponibles, listando su nombre y parámetros.
- `?help command`
  Muestra información sobre los comandos disponibles, mostrando nombre, descripción y descripción de los parámetros.
- `help`
  Muestra la descripción de los ataques, pruebas de vulnerabilidad y pasos de

ataque. Estos mensajes son implementados en cada clase que extienda la clase `Action` implementando el método `getHelpMessage`.

- `connect host port`
  Toma como parámetros el host y el puerto del servidor DLMS/COSEM víctima. Es necesario ejecutar este comando previo a ejecutar cualquiera de los ataques, pruebas de vulnerabilidad o pasos de ataque.
- `list-steps`
  Lista los pasos de ataque disponibles. Toda clase que extienda la clase `AttackStep` será listada.
- `execute-step name params`
  Ejecuta un paso de ataque. Toma como parámetro el nombre del paso de ataque a ejecutar y opcionalmente otros parámetros, dependiendo del paso de ataque en particular. El nombre debe ser ingresado de igual manera que se lista mediante `list-steps`. Retorna la respuesta del servidor.
- `list-attacks`
  Lista los ataques disponibles. Toda clase que extienda la clase `Attack` será listada.
- `execute-attack name`
  Ejecuta un ataque. Toma como parámetro el nombre del ataque a ejecutar. El nombre debe ser ingresado de igual manera que se lista mediante `list-attacks`. Si el ataque requiere el ingreso de otros parámetros, los mismos serán requeridos al usuario. Retorna la respuesta del servidor.
- `list-tests`
  Lista las pruebas de vulnerabilidad disponibles. Toda clase que extienda la clase `VulnerabilityTest` será listada.
- `execute-test name`
  Ejecuta una prueba de vulnerabilidad. Toma como parámetro el nombre del ataque a ejecutar. El nombre debe ser ingresado de igual manera que se lista mediante `list-tests`. Si la prueba requiere el ingreso de otros parámetros, los mismos serán solicitados al usuario. Indica al usuario si la vulnerabilidad está presente o no.
- `vulnerability-check`
  Ejecuta un diagnóstico del servidor. Ejecuta todas las pruebas de vulnerabilidades disponibles. Para las pruebas que requieren el ingreso de parámetros, los mismos serán solicitados al usuario. Como resultado, muestra al usuario cuántas pruebas de vulnerabilidad fueron ejecutadas y cuántas de ellas se encuentran presentes.