ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

SEGURIDAD EN SISTEMAS EMPOTRADOS EN VEHÍCULOS

SECURITY IN VEHICULAR EMBEDDED SYSTEMS

Realizado por
Antonio Acién Gómez
Tutorizado por
Javier López Muñoz
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Septiembre 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: Los automóviles modernos cuentan con una gran cantidad de servicios que suponen potenciales puntos de entrada a ataques exteriores, ya sea de manera física o remota. Cada vez se pueden observar más noticias al respecto y es un tema sobre el que hay un gran desconocimiento, tanto por parte de los usuarios como de los propios fabricantes, que en muchas ocasiones no toman las medidas pertinentes, o lo hacen mal y tarde. Sin embargo, se está tratando de aumentar la seguridad en aspectos muy diversos referentes a los vehículos, teniendo en cuenta las imitaciones en procesamiento, tamaño y coste de sus componentes. Este trabajo pretende llevar a cabo un análisis de las distintas situaciones que pueden poner en compromiso un vehículo y las soluciones que la industria y los investigadores han ofrecido, partiendo además desde los aspectos más básicos, como los buses internos CAN de los vehículos, su formato, manera de comunicación, capas y control de errores.

Palabras claves: seguridad, CAN, vulnerabilidad, ECU, firmware, firewall, cifrado, vehículos

Abstract: Modern cars have a great amount of services that mean potential entry points to external attacks, be it from a physical or a remote access. There are more and more news related to this, and there is a huge lack of awareness about this topic, not only from the users but from the manufacturers themselves as well, who often do not take the pertinent measures, or they arrive late and in a wrong way. Nevertheless, lately there has been a trend in trying to rise the security in several aspects related to the vehicles, taking into account the limitations in processing, size and cost of the components. This work aims to make a critical analysis of the different situations that can compromise a vehicle, and the solutions offered by both investigators and the industry, beginning from the basics, such as the internal CAN buses, their format, way of communicating, layers and error control.

Keywords: security, CAN, vulnerability, ECU, firmware, firewall, encryption, vehicles

Índice

1. Introducción

Los vehículos conectados son una realidad de un tiempo a esta parte. Es un mercado que ha surgido recientemente, en parte por la proliferación de dispositivos como los smartphones que permiten conectarse e interactuar mediante diversos protocolos con distintas partes de los vehículos, añadiendo y simplificando así funcionalidades.

La idea lleva en desarrollo varios años. De hecho, ya en 2005, General Motors anunció la investigación de redes V2V (*Vehicle-to-vehicle*) que permitiesen el intercambio de información[1], lo cual se ha ido concretando en infraestructuras que se usan actualmente. Estos vehículos suelen incorporar una LAN inalámbrica, así como redes propietarias de las marcas que permiten la comunicación para implementar características (Opel OnStar o Chrysler Uconnect son algunos ejemplos).

La información que se comparte con vehículos cercanos nos permite desde conocer si hay atascos hasta prevenir accidentes. Gracias a la integración con *smartphones* podemos conocer y controlar el estado de nuestro vehículo (temperatura, motor…), actuar sobre sus controles de manera remota (activar la climatización, localizarlos, abrirlos…) o incluso manejar el software (reproducción de música o navegación GPS, por ejemplo).

Sin embargo, el coste asociado a estas funcionalidades es una cuestión a tener en cuenta. Se están transmitiendo datos sensibles. Se puede tratar de la localización de nuestro vehículo, el estado de sus puertas, ventanas y luces, o hasta la dirección y los mandos. En estas situaciones, el cifrado y la anonimización de los datos son clave. Y no se trata de escenarios hipotéticos, ya ha habido situaciones reales que ilustran la gravedad de este asunto.

Chrysler tiene un sistema, el antes mencionado Uconnect, que consta de un hotspot Wi-Fi con un rango de 45 metros y con una red móvil 3G. Incluye Bluetooth, satélite de televisión y GPS entre otras características, además de una conexión a internet corriente. Es un sistema por suscripción con una prueba gratuita en los vehículos nuevos. La conexión móvil que provee Uconnect tenía una vulnerabilidad: dejaba total acceso a quien conociese la IP del vehículo. Una vez se tuviera acceso, mediante ataques, se podía acceder a la unidad de cabecera del coche e instalarle el *firmware* malicioso, capaz de enviar comandos a través de la red interna por el bus CAN (*Controller Area Network*) a unidades tan esenciales como el motor o las ruedas. Esto fue lo que hicieron Miller y Valasek, que demostraron ser capaces de detener en seco un Jeep Cherokee en la autovía a 112 km/h[2], además de encender el aire acondicionado, cambiar

de cadena de radio, mostrar las imágenes que deseasen en el ordenador de a bordo y dejar los mandos inutilizables al conductor.

Los dos encargados de explotar esta vulnerabilidad dieron una charla en la conferencia de seguridad Black Hat, haciendo parte de ella pública y trabajando con la propia compañía para solventarla. Esto provocó que se introdujese un proyecto de ley sobre la seguridad en automóviles en el senado de los Estados Unidos. En el sitio web de Chrysler se notificó un parche como actualización de seguridad, el cual debe ser instalado manualmente mediante una memoria USB, lo cual quiere decir que la gran mayoría de coches que no hayan pasado por un concesionario a renovar el software, siguen siendo vulnerables a estos ataques, conocidos y públicos. Esto deja casi medio millón de vehículos circulando con este fallo de seguridad.

Incluso en vehículos que no tienen tantas características de conexión como el Uconnect que nos ocupa, se han dado casos de brechas de seguridad más básicas. Se ha informado de oleadas de robos en Los Angeles y Toronto[3], entre otras ciudades, con dispositivos que abrían las puertas de coches de distintos modelos y marcas (BMW X5, Toyota Prius…). Estos dispositivos son amplificadores de señal, ya que el protocolo de apertura de puertas de algunos coches era tan simple como un desafío-respuesta que simplemente esperaba a escuchar una clave correcta por parte de las llaves, que si estaban emitiendo en un rango suficientemente cercano se podían captar y dirigir al vehículo en cuestión.

Esta clase de fallos también ha afectado al hardware. Las compañías quieren detectar cuando se ha cambiado una pieza relevante por una que no sea de la casa o se ha alterado su funcionamiento. Esto a priori puede no resultar preocupante, pero se puede tratar de partes que afecten al motor o la dirección del vehículo sin estar homologadas o tacómetros en el caso de los transportistas.

Cuando tenemos que lidiar con estos problemas nos enfrentamos a una serie de inconvenientes. Primeramente, en el sistema empotrado de un coche, no tenemos una gran capacidad de cómputo. Las primitivas criptográficas que se usan pueden ser costosas, especialmente si dependen de la longitud de la clave. Otra cuestión es el precio, ya que puede salir muy caro establecer un protocolo totalmente seguro, sin vulnerabilidades para abrir las puertas, cuando alguien podría simplemente romper una ventanilla, teniendo también en cuenta que su diseño encarecería el coste del producto.

También hay que tener en cuenta que no hay aún una concienciación real sobre la seguridad, tanto por parte de la industria como de los usuarios. En algunos estudios sobre la cantidad de dinero dedicada a los diferentes aspectos de los vehículos modernos, ni siquiera es parte integrante del grueso[4].

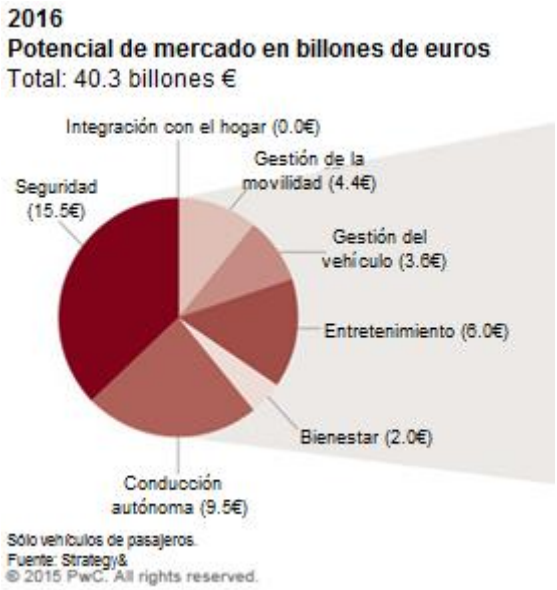


Fig.1: Distribución del mercado de automóviles conectados.

Sin embargo, es innegable que la conectividad está cada vez más integrada en los coches modernos y en nuestra vida. AT&T anunció que en el primer cuatrimestre de 2016 tenía más de 8 millones de vehículos conectados a su red en Estados Unidos (aunque no todos se habían suscrito a sus servicios), contando con acuerdos con grandes fabricantes como Tesla, Ford, GM y Volkswagen[5]. Business Insider estimó que para 2021 habrá más de 380 millones de coches conectados, estando el grueso de ellos en Estados Unidos, Europa Occidental y China, aunque también Brasil, Japón e India integran una buena parte del mercado[6].

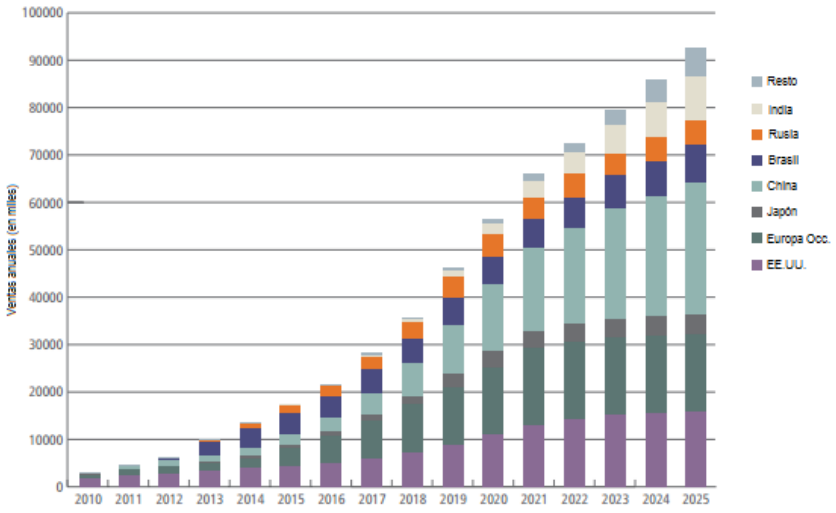


Fig. 2: Crecimiento de las ventas anuales de aparatos telemáticos empotrados segmentados por región.

Aunque los vehículos conectados sean relativamente recientes, hay otro gran número de comunicaciones tomando lugar en ellos, y prácticamente todos tienen vías de entrada vulnerables[7]. Teniendo en cuenta que los modernos tienen más de cien millones de líneas de código y cientos de unidades de control (ECUs), el riesgo se incrementa. Organizaciones como el FBI han declarado que es un riesgo real que los coches sean *hackeados* y usados como armas[8], para llevar explosivos detonantes o como ataques suicidas. Se habla de eventos posibles, aunque aún no probables.

Como podemos ver, el tema que estamos abordando no es un escenario hipotético que no se dé en la vida real, se trata de una realidad cada vez más preocupante, puesto que estas tecnologías se encuentran ya implantadas en una cantidad importante de vehículos y cada vez se extienden más, por lo que supone una fuerte motivación para investigar las causas y soluciones a estos problemas.

A lo largo de este trabajo abordaremos la seguridad en los sistemas empotrados en los vehículos desde los aspectos básicos, como el funcionamiento de sus buses internos, su estructura y comunicación, estudiando cuestiones relacionadas con su seguridad, como los tipos de ataques a los que son susceptibles. Una vez hayamos planteado la problemática veremos qué vulnerabilidades concretas son capaces de explotar estos ataques y qué medidas de seguridad se están tomando para solventarlas.

2. El bus CAN

En este apartado vamos a presentar el protocolo de comunicaciones CAN (*Controller Area Network*, red de área de control), diseñado por Bosch, que se encarga de comunicar varios nodos mediante una topología de bus, y hoy es un estándar (obligatorio por ley) en los vehículos comerciales. Veremos su origen, la motivación para su creación y los beneficios que supuso.

Es importante entender su estructura, cómo se comunican los nodos entre ellos y el formato de las tramas. Como otros protocolos, se corresponde con las capas del modelo OSI, entre las que estableceremos una equivalencia. Por último, veremos algunos mecanismos de control de errores que implementa este estándar.

2.1 Historia

A medida que los vehículos se han ido haciendo más modernos y eficientes, teniendo en cuenta la facilidad del uso para el conductor, el impacto medioambiental y la seguridad de los pasajeros, se han ido controlando más parámetros de los mismos, lo cual se ha traducido en la necesidad de tener una unidad central de control. Por ejemplo, en los coches automáticos el motor envía la velocidad del vehículo a la transmisión, que debe decirle a otros módulos cuándo cambiar de marcha. Conectar todos estos módulos para que se comuniquen entre ellos de forma individual y propietaria se volvió desmesuradamente complejo, por lo que se usó el protocolo de bus CAN[9].

El estándar se presentó en 1986, los primeros controladores por parte de Intel y Philips aparecieron en 1987, pero no se integró en un vehículo hasta 1988. No sólo se redujo el cableado de éste de manera drástica, pesando 45 kilogramos menos que el modelo anterior, sino que los sensores trabajaban de forma bastante más rápida.

CAN es uno de los protocolos clave para los diagnósticos de a bordo (OBD, *On-Board Diagnostics*), que son obligatorios desde 1996 (OBD-II) en Estados Unidos y desde 2001 en la Unión Europea (EOBD).

2.2 Estructura básica

La topología del bus CAN es mantener una línea central principal a la que van conectados los demás módulos o ECUs (*Electronic Central Unit*)[10]. Cada uno de los módulos puede ser un sensor, un actuador, o incluso una puerta de enlace para que se comunique otro dispositivo, como un puerto USB o Ethernet. Los nodos se conectan uno a otro por un bus de dos cables y cada uno requiere una CPU, un controlador CAN (normalmente integrado) y un transceptor (para recibir y transmitir, convierte el flujo de información del bus a niveles del controlador y viceversa). Si alguno de los nodos falla, el sistema sigue funcionando, a menos que haya módulos directamente dependientes, lo cual lo hace más seguro. También simplifica el sistema de control y hace que sea mucho más sencillo introducir módulos nuevos. Cualquier módulo puede alertar al controlador de la ocurrencia de un evento, es decir, puede enviar información, así como recibir, pero no simultáneamente. Hay dos tipos de bus: el de alta velocidad, que usa un único bus lineal, y el de baja velocidad o tolerante a fallos, que puede usar uno lineal, en estrella, o múltiples en estrella conectados por uno lineal.

2.3 Comunicación

En la especificación de CAN se define como “dominante” y “recesivo” (palabras que provienen de la terminología genética) al 0 y 1 lógico, respectivamente. Además, hay cuatro tipos de trama CAN: de datos, de error, remota y de sobrecarga, cada una con sus características, que se describen a continuación.

Las tramas de error contienen un flag que indica el tipo de error (activo o pasivo), además de los datos del mismo.

Las tramas de sobrecarga se pueden interpretar como tramas de error que no obligan al emisor a realizar una retransmisión.

Las tramas remotas se usan cuando un nodo necesita información de otro para continuar su funcionamiento. Esto sucede bastante a menudo, porque los actuadores necesitan conocer el estado de los sensores para poder realizar su labor en consecuencia.

Hay dos formatos de la trama de datos, con ID básico (11 bits) y extendido (29 bits), que lo transmite dividido en dos partes (11 y 18 bits). La ventaja de este último es poder enviar un número mayor de tipos de mensaje, sacrificando tiempo de acceso al bus y por tanto, rendimiento. Por tanto se recomienda usar el formato estándar siempre que sean suficiente 2032 tipos distintos de mensajes (serían 2048, pero por detalles de implementación algunos de ellos están reservados). En cualquier caso, los controladores que soportan el formato extendido tienen retrocompatibilidad con el estándar, pueden coexistir en el mismo bus y no añaden carga a la CPU, puesto que de la capa física y de transporte de CAN se ocupa casi exclusivamente el hardware.

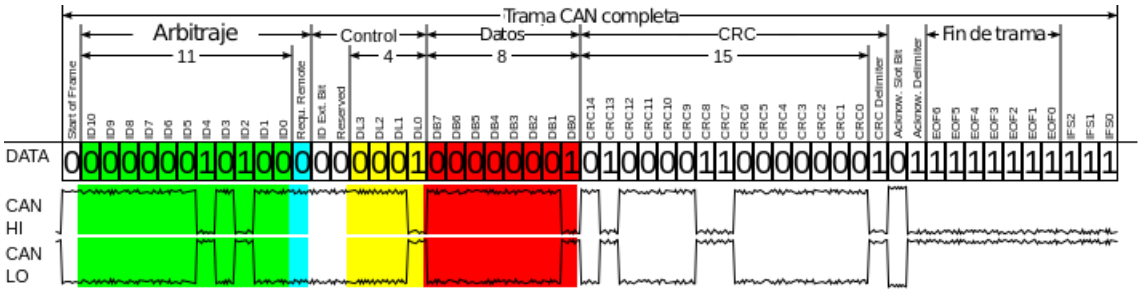


Fig. 3: Trama CAN (formato estándar).

Campo	**Longitud (bits)**	**Propósito**
Comienzo de la trama	1	Denota el comienzo de la transmisión
ID	11	Identificador único que indica la prioridad
Petición de transmisión remota	1	0 para tramas de datos, 1 para tramas de petición remotos
Bit de extensión del ID	1	0 para las tramas con identificador básico de 11 bits
Bit reservado	1	Según la especificación debe ser 0, pero no importa
Código de longitud de datos	4	Número de bytes de datos
Campo de datos	0-64	Datos transmitidos
Código de redundancia cíclica	15	Comprobación de errores
Delimitador CRC	1	Debe ser 1
Hueco ACK	1	El módulo que transmite envía 1 y cualquier receptor puede poner 0
Delimitador ACK	1	Debe ser 1
Fin de trama	1	Debe ser 1

	Formato estándar	Formato extendido
Número de IDs	O	+
Tiempo de acceso al bus	+	O
Rendimiento del bus	+	O
Carga de CPU	+	+
Disponibilidad de productos	+	-
Relación tamaño chip / coste	+	O
O – *Media* **|** + – *Por encima de la media* **|** - – *Por debajo de la media*		

Fig. 4: Comparación entre formatos de tramas CAN.

La comunicación de tramas dentro del bus se hace con *broadcast*, es decir, la trama viaja por el bus sin un destinatario concreto, los nodos no tienen direcciones, sino que todos reciben cada trama y deciden si descartarla o actuar.

2.4 Capas

Podemos adaptar el protocolo CAN al modelo OSI, y se ocuparía del nivel físico y de enlace. Pasamos a describir esta equivalencia y de las tareas que se encarga cada uno según este esquema.

2.4.1 Nivel físico

CAN usa como infraestructura física un par trenzado de cables, CAN_H y CAN_L, cada uno con una resistencia de 120 ohmios. Los estados dominante y recesivo que hemos ilustrado anteriormente se definen como una diferencia de voltaje entre ellos mayor o menos que el umbral mínimo (<0.5V en la entrada del receptor o <1.5V en la salida del transceptor)[11].

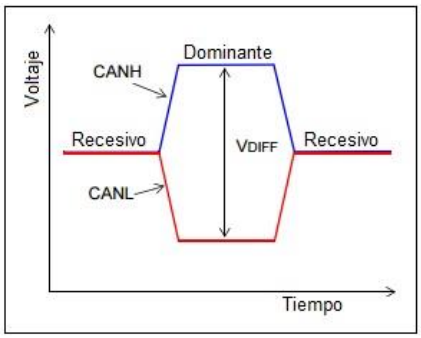


Fig. 5: Ilustración de los estados dominante y recesivo lógicos según voltaje físico.

El estándar también indica que a cada extremo del bus debe ir una resistencia nominal también de 120 ohmios, para evitar ondas reflejas debido a impedancias incorrectas en los nodos del bus, o bien dos resistencias de 60 ohmios con un condensador de acoplamiento, para filtrar componentes de alta frecuencia en el bus[9].

A nivel físico CAN requiere que todos los nodos estén sincronizados con una señal a nivel de bit, dada la codificación de bits que se usa en el canal (NRZ, *Non Return to Zero*), de manera que no hay ninguna señal entre ellos cuando hay varios bits iguales seguidos transmitiéndose. Esta sincronización empieza con el bit SOF (*Start of Frame*, comienzo de trama), sin embargo, debido a retrasos o desajustes de frecuencia se puede perder, por lo que hay una resincronización constante para la cual las ECUs siguen un protocolo usando los flancos de las transiciones. En caso de que no hubiera flancos en más de cinco bits transmitidos se usa un “bit de complemento” o *padding* para resincronizar, ya que muchos ciclos de reloj sin flancos aumentarían las posibilidades de que la sincronización se pierda. Estos bits son insertados por el nodo que transmite y se ignoran al ser recibidos. Esta sincronización hace posible que los mensajes CAN sean más simples.

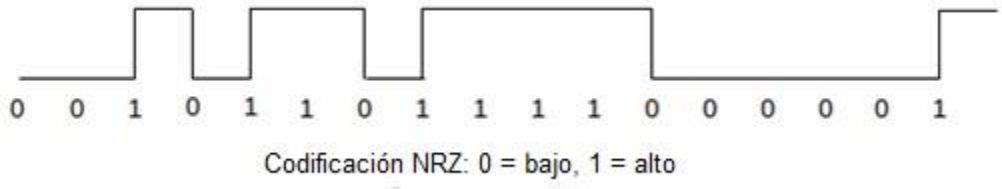


Fig. 6: Codificación de bits mediante NRZ.

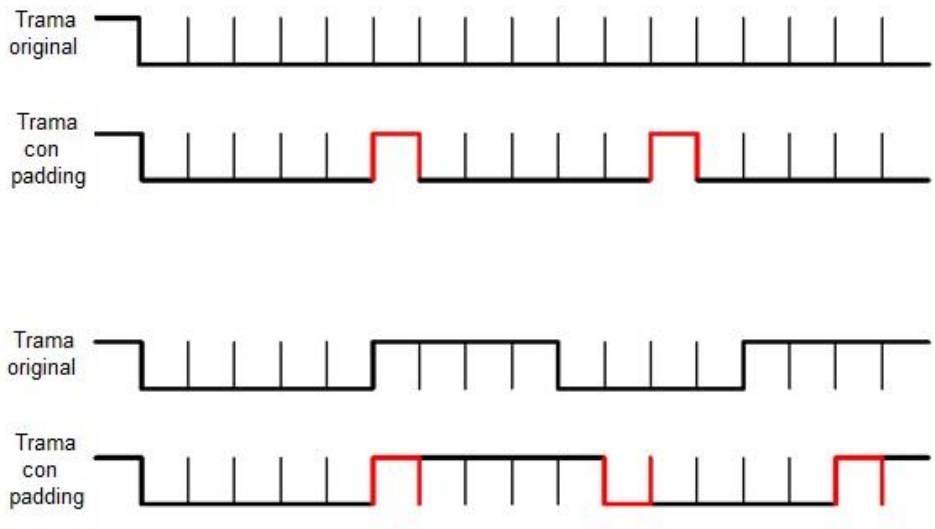


Fig. 7: *Padding* de bits en el bus CAN.

Para el arbitraje y el control de errores, el tiempo de transmisión de un bit tiene que ser al menos suficiente para que la señal se propague de cualquier emisor al receptor y de vuelta. En la práctica se determina con la suma de varios segmentos, teniendo en cuenta el tiempo de la sincronización, el de propagación por el bus y el retraso del transceptor de cada nodo en traducir los bits a una instrucción que entienda el controlador CAN[13].

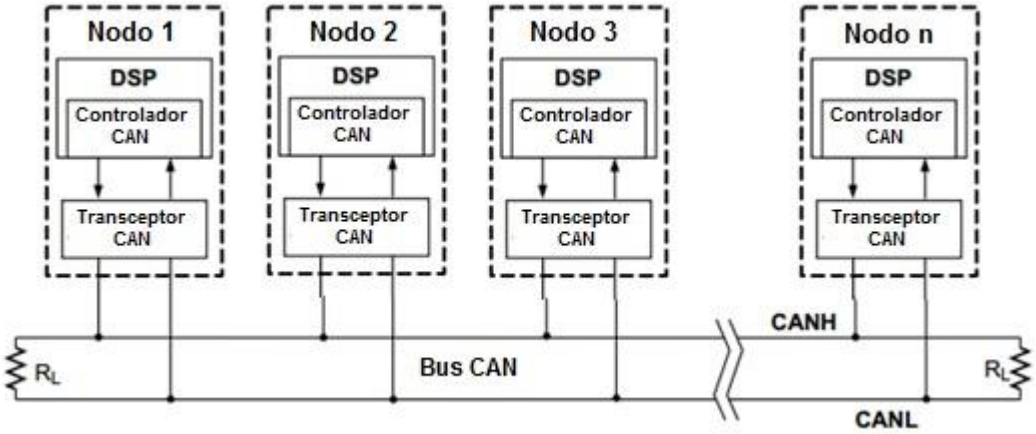


Fig. 8: Topología de un bus CAN y sus nodos. (DSP = *Digital Signal Processor*)

2.4.2 Nivel de enlace

El grueso de la especificación de CAN se refiere a esta capa, involucrando funciones tales como detección de errores, validación de mensajes, mensajes ACK y arbitraje. También se ocupa del encapsulado del mensaje en tramas.

La etapa de arbitraje tiene el cometido de decidir qué módulo transmitirá. El estado “ocioso” se representa con el 1 lógico. Si un nodo transmite un bit dominante y otro uno recesivo hay una colisión, y el dominante gana el arbitraje, así no hay retraso en el envío del mensaje de prioridad más alta, mientras que el que ha perdido el arbitraje intentará volver a transmitir el mismo mensaje seis ciclos de reloj más tarde. Este sencillo mecanismo hace posible que la comunicación sea en tiempo real pero con control de prioridad en los mensajes.

CAN necesita que todos los nodos, incluyendo al emisor del mensaje y al receptor, estén escuchando el bus. Si todos los nodos que estén transmitiendo envían un 1, todos los nodos ven un 1 en el canal. Si todos envían un 0, todos ven un 0 en el canal. Sin embargo, si algunos envían un 1 y otros un 0, todos ven un 0, incluyendo los que envían un 1, los cuales al detectar la disputa dejan de transmitir y lo vuelven a intentar más tarde, como hemos indicado. De este modo, el primer nodo que transmita un 1 pierde la etapa por ser menos prioritario.

Esto funciona porque el ID de cada mensaje transmitido es único y representa su prioridad.

En esta capa también entraría la confirmación de mensajes (ACK) y una función llamada “confinamiento de errores”, que consiste en que un nodo puede dejar de transmitir o apagarse totalmente si se detectan errores suficientes, para evitar fallos catastróficos. Cada nodo tiene dos contadores para los errores: El contador de errores transmitidos (TEC) y de errores recibidos (REC). Se incrementan cuando se transmite o detecta un error y se decrementan cuando hay una transmisión con éxito, y aquí es donde entran en juego los *flags* de las tramas de error. Si se envía un *flag error-active* significa que el nodo puede seguir participando en las comunicaciones sin restricción, cuando el TEC y el REC se encuentran por debajo de 128. Si cualquiera de los dos pasa de este número, se envía el *flag error-passive* y el nodo deberá esperar ocho bits para poder retransmitir una trama de datos con error. Si el TEC pasa de 255, el nodo entra en estado *bus-off*, que sólo se da por fallos de transmisión, al ocurrir 32 mensajes consecutivos erróneos. El nodo dejará de recibir o enviar mensajes de cualquier tipo. Se puede reiniciar el controlador, ya sea por hardware o por software, si bien en este último caso se deberá esperar un tiempo determinado. Una vez se reactive el nodo, los contadores estarán a cero y el estado será *error-active*. Se puede expresar todo esto mediante una máquina de estados. Los *flags* de errores activo y pasivo se representan mediante seis bits consecutivos dominantes y recesivos, respectivamente.

Fig. 9: Máquina de estados de error de los nodos CAN.

En las tramas hay varios bits fijos que deben tener un cierto valor, como que el bit SOF debe ser dominante, lo cual es útil, entre otros métodos, para la detección de errores.

También contienen el CRC, que es el resultado de aplicar una ecuación polinómica a la carga de datos transmitida. Al llegar la trama a un nodo se vuelve a aplicar la ecuación y se compara con el CRC. Si no coinciden, se envía una trama de error CRC al emisor, ya que se ha violado la integridad del mensaje.

Puede darse un error de ACK si el nodo espera una confirmación en ese hueco. Si no la recibe o no es el valor que espera, seguirá enviando la trama hasta que la obtenga.

Además, como hemos discutido antes en los bits de *padding*, si se reciben más de cinco bits idénticos seguidos, se sabe que se trata de un error, puesto que esto se evita para realizar la resincronización de las ECUs. La única excepción es con los *flags* de error, por lo que al detectar esto, los demás nodos envían una señal de “eco de error”.

En la subcapa de control de enlace lógico podemos situar el filtrado de mensajes. El ID de las tramas CAN no indica nada sobre su destinatario, así que cada nodo se encarga de tomar la decisión de descartarlas o procesarlas. Esta subcapa también es la encargada de intentar transmitir de nuevo un mensaje si pierde el arbitraje.

Como podemos ver, la arquitectura del bus CAN es bastante simple, ya que requiere comunicaciones rápidas para elementos vitales del vehículo y además se basa en componentes empotrados, que no tienen una gran capacidad de procesamiento.

CAN se diseñó con tolerancia a fallos, es sencillo detectar tramas erróneas y descartarlas[14], pero se diseñó teniendo en cuenta un ambiente cerrado, por lo que las medidas de seguridad ante amenazas externas están ausentes.

3. Seguridad y tipos de ataques

Como CAN no se diseñó teniendo en cuenta un escenario en el que los aparatos que lo implementen estuviesen abiertos a amenazas externas, la seguridad se ha de implementar de manera externa al estándar del protocolo. Aquí veremos qué aspectos se pueden comprometer de las comunicaciones de los vehículos y qué ataques concretos son capaces de explotarlas. Una vez estén definidas, veremos qué maneras hay de acceder a ellos.

Hay diferentes aspectos de la seguridad que debemos tener en cuenta, ya que el tipo de ataques y de vulnerabilidades es muy amplio. Nos referiremos al aspecto software, ya que también podríamos considerar un fallo de seguridad, por ejemplo, la autenticidad de las piezas (si alguien utiliza un repuesto no oficial que pudiera comprometer la seguridad del automóvil). Los diferentes aspectos que se pueden ver comprometidos en las comunicaciones de los vehículos son los que aquí se describen brevemente:

- **Confidencialidad:** Los mensajes que se transmiten únicamente deberían ser vistos por las ECUs a las que atañen. Esto es de especialmente difícil consideración, teniendo en cuenta que la comunicación en CAN se hace con *broadcast* y las ECUs lo ignoran o no en función del identificador.
- **Integridad:** Es importante asegurarse de que un mensaje sigue intacto desde su emisión hasta su recepción, ya que puede haber cambios en él. CAN usa CRC para asegurarse de que se cumpla, pero la implementación de CRC-16, según la documentación oficial[15], en datos de 8 bytes podría causar colisiones, por lo que no es un método lo suficientemente seguro.
- **Disponibilidad:** Los datos en la red deben estar disponibles en cualquier momento. Esto es particularmente difícil, ya que los ataques de denegación de servicio en CAN son sencillos de realizar, y la función de confinamiento de errores permite apagar una ECU en caso de error, por lo que sus datos no estarían accesibles en ese caso.
- **Autenticidad:** Verificar el emisor de un mensaje es importante, ya que un nodo malicioso podría inyectar mensajes fingiendo ser otra ECU, ya que no tienen dirección.
- **Temporalidad:** Dado que los mensajes en CAN no tienen *time-stamping*, alguien podría capturar mensajes durante un tiempo y volverlos a enviar más tarde, aún siendo válidos.
- **No-repudio:** Hay que diferenciar entre un fallo de una ECU en la transmisión de un mensaje o del canal, y un mensaje malicioso enviado a propósito. Si los nodos ignoran todo lo malformado y no se retransmite podrían pasarse por alto mensajes importantes.

Cada uno de estos aspectos de la seguridad tiene ataques asociados: lectura de paquetes (confidencialidad), inyección de paquetes (autenticidad), modificación de paquetes (integridad), denegación de paquetes (disponibilidad) o inundación (denegación de servicio), *replay* (temporalidad, autenticidad…), entre otros. Por supuesto, estos ataques se pueden combinar entre ellos

En los vehículos actuales, aunque CAN es el estándar que siguen los buses *event-triggered* (que comienzan su funcionamiento cuando hay un evento), hay más buses con los que comparten comunicación mediante puertas de enlace. Según su función, implementan distintos protocolos, normalmente los que se describen a continuación:

- **LIN:** es una red que provee la comunicación entre los sensores y los actuadores en sistemas más pequeños (control de espejos o ventanillas y cierre de puertas, por ejemplo). Tiene un sólo nodo maestro y hasta 16 esclavos, y sigue un mecanismo de hibernación, comienza su funcionamiento cuando le llega un paquete de *wake-up* para reducir su consumo. Para controlar errores tiene mecanismos como bits de paridad y checksums.

- **FlexRay:** tiene un alto ancho de banda para comunicación a alta velocidad y sigue un esquema de bus como CAN con hasta 64 nodos. Usa TDMA (*Time Division Multiple Access*) tanto en modo síncrono como asíncrono y tiene un protocolo de checksum para detectar errores.

- **MOST:** se usa para transmitir audio, vídeo y flujo de datos de control. Es de alta velocidad y está diseñado para buses que tengan una dirección para el emisor y otra para el receptor.

También hay que indicar que los automóviles no suelen tener un solo bus CAN, sino uno para las tramas de alta prioridad (sistemas de dirección, freno, motor…) y otro para el funcionamiento más secundario (radio, ventanas, entretenimiento…).

Todos estos sistemas de comunicación son susceptibles a ataques, ya que entre ellos hay puntos de enlace, por lo que si presentan vulnerabilidades, el sistema entero está comprometido.

Sabiendo la estructura y el funcionamiento básico de los buses internos del vehículo, y las maneras en las que se puede ver comprometida la seguridad, podemos ver las distintas formas en las que un atacante con acceso al vehículo podría alterar su actividad normal o esperada.

- **Firmware malicioso:** Con ligeras modificaciones (poco más de 100 líneas de código) se puede hacer, por ejemplo, que los frenos se desactiven o las luces se apaguen si el coche pasa de una cierta velocidad[16]. También hay software especializado para modificar tanto el

firmware de las ECUs como sus variables calibrables (por ejemplo, parámetros de inyección del motor), típicamente usado en concesionarios.

El riesgo potencial es altísimo, dado que muchos modelos automóviles permiten *flashear* las ECUs para instalar firmware distinto sin necesidad de autenticación, con un método de desafío-respuesta que tiene sus claves *hardcodeadas* (incluídas directamente en el código del programa) o cuyo resultado no se verifica.

Y lo que es aún peor, se ha podido *flashear* las unidades mientras se conducía el vehículo. Esta violación de la autenticación y la posibilidad de poner las ECUs en modo *reflash* mientras el coche esté en marcha contradicen el estándar de CAN, que señala claramente que “el módulo de control del motor debería rechazar la petición para iniciar un evento de programación si el motor estuviese encendido”.

Otro aspecto preocupante de la facilidad de instalar software malicioso es que se podría programar para borrarse a sí mismo después de efectuar su ataque, eliminando prácticamente cualquier posibilidad de rastreo de manera muy sencilla, en algunos casos, simplemente reiniciando la unidad telemática del vehículo[16].

- **Flooding:** Dada la sencilla etapa de arbitraje, si se inunda el bus con bits dominantes, ninguna de las ECUs podría transmitir. Para solventar esta problemática se han intentado introducir filtros de tiempo que limiten la tasa de mensajes transmitidos de una interfaz a otra y evitar la denegación de servicio[16].

- **Mensajes maliciosos:** La cantidad de mensajes válidos que se pueden enviar es CAN es limitada, por lo que es relativamente sencillo detectar tramas maliciosas. También es una manera de filtrar la mayoría de mensajes que se envíen mediante *fuzzing* (envío de datos aleatorios).

  Mediante esta técnica se han logrado controlar funciones clave del vehículo, tales como la apertura y el cierre de las puertas, el ajuste de las luces, las ventanillas o los limpiaparabrisas[16].

  Algunas compañías han intentado solucionar esto introduciendo un *firewall* en el bus[17] que inspecciona el identificador y la carga de datos de cada una de las tramas y las compara con unas *white-list* (dependiendo del bus CAN en por el que se estén transmitiendo) que contiene reglas con los mensajes seguros para cerciorarse de que no provoquen un mal funcionamiento.

  Por otra parte, si no hay manera de garantizar que los mensajes sean recientes, tampoco hay manera de asegurar que cada trama que llegue no sea un *replay* de una capturada anteriormente.

- **Falta de anonimato:** Muchos de los datos que se recaban en los vehículos pueden ser capturados para estadísticas, ya sea por parte del fabricante con autorización (en muchos casos muy ambigua), o para conocer datos personales por parte de terceros sin ella[18]. Estos datos pueden ser simples como el estado de la batería para asistencia remota, pero también se sabe que se puede enviar una alerta si el vehículo sale de ciertos límites geográficos, si se arranca después de cierta hora, si se pasa de un límite de velocidad, o actuar en el coche a distancia (abrir y cerrar las puertas o encender y apagar las luces).

Todos estos posibles ataques necesitan un vector de entrada para poder comprometer al vehículo. Como hemos visto, los vehículos modernos cuentan con una gran superficie susceptible a ataques, cada uno con sus vulnerabilidades. Ya que los vehículos actuales están obligados por la legislación a llevar un puerto OBD-II para el diagnóstico, es sencillo usar un *dongle* o algún otro dispositivo que se conecte a ellos. En los vehículos modernos, el acceso está enfocado a hacerse desde el PC, y por ello, desde 2004 en Estados Unidos, los vehículos están obligados a incorporar soporte de PassThru, una API que provee una manera sencilla de comunicarse con los buses internos. El dispositivo PassThru se comunica con un PC, ya sea cableado o de manera inalámbrica, expandiendo así la superficie vulnerable, ya que, como veremos, puede ser víctima de muchos fallos de seguridad.

El conector OBD por su parte tiene un esquema de 16 pines, estando definidos los 6 y 14 como *CAN-High* y *CAN-Low* respectivamente (a veces indicados como CANH y CANL), pudiendo usarlos de entrada. No hay manera de saber si estos pines se usan de sólo salida, ya que la implementación queda a cargo del fabricante. Teniendo acceso a él es sencillo *flashear* el firmware de cualquier ECU, instalando uno malicioso, como hemos visto anteriormente, por lo que supone un grandísimo riesgo.

Hay más aún vías de entrada físicas que están abiertas a medios externos. Desde los reproductores de CD se puede comprometer una unidad de entretenimiento y obtener acceso a las demás. Muchos modelos nuevos además cuentan con conexiones para USB o iPod/iPhone, y también hay tecnologías más sencillas que cuentan con vulnerabilidades, como la apertura remota con llave electrónica. Cada nueva característica supone un gran número de potenciales amenazas que se han de controlar en el firmware.

En el caso de ataques avanzados, si se tiene acceso físico al vehículo, las ECU se pueden desmontar y extraer piezas de ellas. Hacer un volcado de memoria de los discos flash con los que cuentan a otro medio físico y examinarlos ha sido llevado a cabo con éxito, y gracias a ello se puede acceder a claves criptográficas, certificados y a ficheros binarios y scripts que pueden dar idea del funcionamiento interno del sistema[19]. Si las claves están en posesión de un atacante, podría iniciar sesión en una terminal por SSH en la unidad telemática

y con acceso *root*, ejecutar comandos arbitrarios y acceder a cualquier archivo. Es especialmente preocupante que se ha visto que en ciertos modelos la clave viene por defecto en todas las unidades del mismo, pudiéndose acceder a todos los coches de esa misma gama con una única clave.

Sin embargo, los riesgos que más preocupan son los que se hacen mediante acceso remoto, ya que por ejemplo si se tratase de un robo, cuando se puede estar lo suficientemente cerca físicamente del vehículo se podrían llevar a cabo ataques mucho menos sutiles (no merece la pena las horas y la investigación para comprometer el sistema de bloqueo de las puertas cuando simplemente se podría forzar una puerta)[20].

En la superficie de ataque remota podemos establecer la distinción en el rango de alcance de las tecnologías, teniendo las de rango corto por un lado, y las de largo por otras.

En el rango inalámbrico corto podemos ver cómo se pueden realizar ataques a través de Bluetooth, ya que es el estándar de facto para llamadas manos libres, así como sincronización con distintos dispositivos móviles para música o distintas aplicaciones. Estos dispositivos se podrían comprometer, o podrían ser simplemente vulnerabilidades en la implementación del protocolo lo que dé lugar a agujeros de seguridad.

Muchos vehículos modernos también incluyen un hotspot Wi-Fi para que se conecten dispositivos que a su vez podrían estar comprometidos.

El aspecto más preocupante, por supuesto, es cuando entran en juego conexiones móviles. Los vehículos con una conexión 3G y de voz asociada tienen una gran cantidad de servicios. Por ejemplo, el servicio OnStar de Opel/GM cuenta con localización GPS, avisa en caso de emergencias (incluso establece comunicación telefónica entre los pasajeros y un equipo que les atienda), avisa de adversidades en la vía como accidentes o atascos, conecta con otros vehículos para intercambiar información, e incluso, en ciertos modelos, puede detenerlos de manera remota en caso de robo. Para llevar a cabo la mayoría de estas tareas se requiere una fuerte interconexión entre los buses internos del vehículo. Conexión que, además, debido a su naturaleza, ha de poder ser accedida remotamente desde cualquier punto y en cualquier momento (siempre que tenga cobertura), y por tanto, sus vulnerabilidades se podrán explotar a través de la red. Ataques como parar completamente el motor de un vehículo en la autovía desde un portátil en un domicilio ya se han llevado a cabo. Así entran en juego tanto los fallos de seguridad de las propias ECU como las implementaciones de las redes móviles en el firmware de los coches.

4. Análisis de vulnerabilidades

En la siguiente sección vamos a exponer ataques llevados a cabos con éxito en diversas investigaciones. Para no poner en el peligro la seguridad de los conductores, en la mayoría de ellas no se menciona el modelo exacto en el que se llevaban a cabo, pero se trata de vehículos de gama media que llegaron al mercado en los últimos diez años. Para cada una de las vulnerabilidades encontradas se ha logrado hallar la manera de comprometer el vehículo en su totalidad, accediendo a partes vitales como el motor o la dirección. Dividiremos esta parte según la vía de entrada que se haya usado para acceder al vehículo.

4.1 Reproductor de CD

Una de las vulnerabilidades más flagrantes encontradas ha sido la de algunos reproductores de CD[21]. Al hacer ingeniería inversa y observar el código fuente, se podía apreciar cómo algunos modelos aceptan un archivo con un nombre específico. Al encontrarse éste presente, se muestra un mensaje críptico en la pantalla, y si el usuario no pulsa el botón correspondiente, se flashea el *firmware* del reproductor con el contenido de este archivo. Programar un *firmware* malicioso es una tarea que requiere tiempo y conocimiento, pero aun así es un riesgo muy a tener en cuenta. Esta función probablemente se conserve por retrocompatibilidad con *firmware* antiguo.

Los reproductores de CD además tienen la capacidad de *parsear* archivos complejos, por lo que se puede codificar un archivo MP3 o WMA que sea capaz de enviar mensajes maliciosos. En la implementación de los códecs WMA se observó que se hacían suposiciones sobre la duración de los archivos de entrada, y la implementación del *parser* permite hacer lecturas de longitud arbitraria. Combinando estos dos fallos se produce un *buffer overflow* del que no es trivial sacar partido, ya que el desbordamiento se produce en el BSS, una parte del segmento de datos que contiene variables alojadas estáticamente que inicialmente tienen asignado como valor cero. Tras el BSS hay varias variables dinámicas que se comprueban continuamente y hacen que el sistema se cuelgue si se sobreescriben arbitrariamente. Con la ayuda de un *debugger* que permita poner *breakpoints* y hacer volcados de memoria, se puede determinar qué segmentos de memoria que contengan funciones o variables se pueden sobreescribir sin que el funcionamiento normal del reproductor se vea afectado. Así, se puede codificar un archivo que en un PC funciona a la perfección pero que en algunos reproductores empotrados enviará mensajes CAN maliciosos. Este tipo de archivos se podría extender fácilmente, por ejemplo en redes P2P, porque es prácticamente indistinguible de un cualquier otro archivo de audio.

4.2 Dispositivos PassThru y OBD-II

Como hablamos en apartados anteriores, en Estados Unidos es obligatorio el estándar PassThru en los vehículos desde 2004 para permitir el diagnóstico y *flasheado* de sus unidades internas desde un PC, ya sea de manera cableada o inalámbrica.

Los dispositivos PassThru tienen una arquitectura con un procesador *System-on-a-chip* y cuentan con varias interfaces de red, y como cualquier otro elemento están expuestos a vulnerabilidades. En algunos modelos se observaron las siguientes.

Cuando el dispositivo PassThru se enciende, anuncia su presencia con un paquete multicast UDP, indicando a qué red se conecta, su IP y el puerto TCP en el que escucha las peticiones de los clientes. Los clientes se conectan al puerto y configuran la conexión, pero la comunicación entre la aplicación del cliente y el dispositivo es sin autenticar. Cualquier seguridad que se quiera implementar la tiene que hacer la red de manera externa, si bien dos clientes no se podrían conectar a la vez al dispositivo. Con cualquier *packet sniffer* se podrían ver las comunicaciones y conocer los detalles del firmware que se esté instalando o las comprobaciones que se estén realizando, información que para mayor seguridad debería ser anónima.

También se descubrió que la API (que sirve para configurar el estado del dispositivo PassThru y sus parámetros de red, entre otras cosas) tenía bugs que permitían ejecutar comandos mediante *shell-injection*, contando el kernel Linux del aparato con programas como *telnet*, por lo que se podrían abrir conexiones y transferir datos o código por ellas. Este bug es especialmente peligroso, ya que los PassThru se conectan a una red en la que probablemente haya más (piénsese en un concesionario o taller). En la Universidad de California y San Diego explotaron esta vulnerabilidad diseñando un programa que entraba a *telnet* de cada PassThru que se anunciaba con el paquete multicast e instalando en ella software malicioso, de manera que se extiende a los demás dispositivos PassThru de la red cuando los detecta, además de transferir software infectado a las ECU de los vehículos.

4.3 Bluetooth

Normalmente las funciones Bluetooth y la implementación del protocolo van en la unidad telemática del vehículo, la cual lleva incorporado un firmware que en ciertos vehículos ha demostrado ser susceptible a ataques. Haciendo ingeniería inversa al software basado en Unix se descubren fallos como llamadas a *strcpy* sin ningún tipo de control. Esto quiere decir que desde cualquier dispositivo pareado se podían enviar cadenas de longitud arbitraria, ya que ésta no se

comprobaba realmente, e incluir en ellas código para ser ejecutado mediante inyección.

Para explotar ciertos aspectos se necesita un dispositivo móvil previamente pareado con la unidad del vehículo. Esto es un obstáculo, pero para un atacante con los conocimientos necesarios no supone un gran problema comprometer un móvil con troyanos o similares. Se puede desarrollar uno que monitorice conexiones Bluetooth y si ve una unidad telemática, envíe como carga en el mensaje el código malicioso.

Sin embargo, se pueden realizar ataques sin este requisito, conociendo la MAC del vehículo y teniendo un dispositivo propio para parearlo. La dirección MAC se puede obtener haciendo *sniffing* de paquetes si el coche está arrancado en presencia de otro dispositivo pareado, y una vez la tengamos, se necesita tener un secreto compartido (PIN) entre el vehículo y nuestro dispositivo. Normalmente, éste se genera aleatoriamente y se muestra en el panel de mandos, teniendo que introducirlo el usuario manualmente en su terminal, pero la implementación de la pila de protocolo Bluetooth en algunos vehículos responde automáticamente a las peticiones de pareado, sin necesidad de interacción del usuario siquiera, por lo que realizar un ataque por fuerza bruta es simple, y aunque consuma tiempo, es paralelizable (podría estar haciéndose a varios vehículos al mismo tiempo). Una vez se efectúe con éxito, podremos inyectar código tal y como se comenta en el párrafo anterior. Este ataque puede parecer una tarea ardua, especialmente porque la unidad telemática del automóvil ha de estar encendida, pero si se efectúa en un parking o cualquier otro entorno en el que haya una cantidad considerable de vehículos, enviando un mismo PIN a varios, aumentan las probabilidades de que algún coche responda satisfactoriamente. En pruebas empíricas, se tardó en lograr el pareado con éxito desde quince minutos hasta trece horas y media.

4.4 Redes móviles

Como hemos detallado anteriormente, el uso de comunicaciones móviles tiene grandes ventajas en la conducción, pero conlleva riesgos asociados.

Las unidades telemáticas suelen llevar asociados servicios de 3G para conexión de datos, de voz (útil cuando no hay cobertura 3G en muchos lugares) y de SMS, que se usa para activación de servicios en algunos casos.

Si el atacante conoce el número del vehículo objetivo puede llevar a cabo estos ataques de manera sencilla. Sin embargo, hay maneras de obtener los números de potenciales objetivos. En Estados Unidos se observó que el prefijo que tenían asignado era 566 (reservado para dispositivos de comunicación personal), y a partir de un número concreto de un vehículo, era probable que los números colindantes también perteneciesen a otros, lo cual se podía descubrir enviando

un SMS de *status* pidiendo su estado, ya que como veremos más adelante, poseen una interfaz SMS desde la que se puede obtener información[19].

4.4.1 Conexión de datos a través de llamada

En Norteamérica el uso de aqLink para usar una llamada como una conexión de datos está ampliamente extendido. Esta conexión de datos se suele hacer entre el vehículo y un centro telemático de la compañía (o de una empresa que ofrezca estos servicios)[22] usando un simple protocolo de tonos de llamada para pasarla a modo de datos. Usando ingeniería inversa en este protocolo se puede descubrir el patrón con el que se inicia el modo de datos, la frecuencia de los tonos y la codificación de los datos transmitidos.

De esta manera, estudios observaron que el protocolo aqLink soporta paquetes de 1024 bytes, pero asume que la carga nunca excederá los 100 bytes, ya que los mensajes con un formato correcto siempre son de una longitud menor. Esta suposición da lugar a una vulnerabilidad por *overflow*.

Por otro lado, también hay problemas de autenticación. Al iniciarse la llamada, el vehículo envía un desafío aleatorio de tres bytes y se inicia un temporizador. La central telemática hace un *hash* (función que convierte una entrada en una representación compacta dentro de un rango finito, de la que no se puede volver a obtener la original) del desafío con una clave previamente compartida como respuesta al mismo. El vehículo mientras tanto no acepta ningún otro paquete, y si se cumple el temporizador o se recibe una respuesta incorrecta, envía un paquete de error. Cuando se recibe el "ACK" de éste, la llamada se cuelga.

A primera vista es un protocolo bastante seguro, pero el error reside en la generación de los tres bytes aleatorios del desafío, ya que la semilla que se usa para generarlos es constante (sólo se reinicia con la unidad telemática). Esto significa que varias llamadas esperan la misma respuesta correcta al desafío, por lo que se puede hacer un ataque de replay si se ha hecho captura de paquetes anteriores.

4.4.2 SMS

Algunos coches incluyen una interfaz SMS mediante la que se puede interactuar. Una vez conozcamos el número de teléfono asociado a la unidad telemática se le pueden enviar mensajes de texto que devuelvan valores de los sensores, posición de GPS, actualizaciones remotas…

Es el caso de las actualizaciones remotas el más peligroso, ya que el método que se sigue es obtener un archivo mediante el protocolo SCP del servidor de actualizaciones. Este archivo, al que llamaremos *UpdateFile*, contiene nombres de otros archivos, sus rutas y sus *hash*, ya vayan a ser añadidos o borrados. Si alguno de estos archivos no está presente o su *hash* es incorrecto, se obtiene

por el mismo método del servidor. Finalmente, si el archivo de actualización contiene algún comando, se ejecuta.

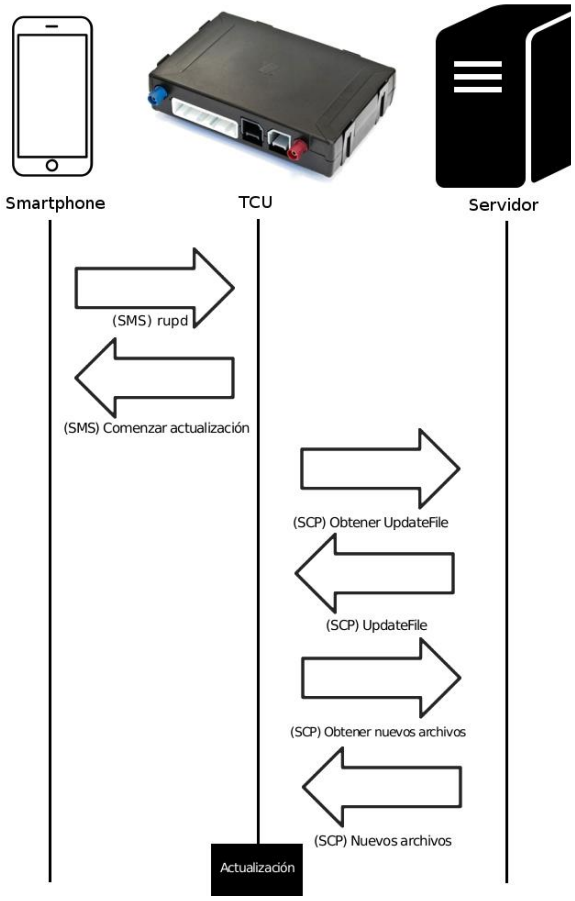


Fig. 10: Actualización vía SMS del firmware.

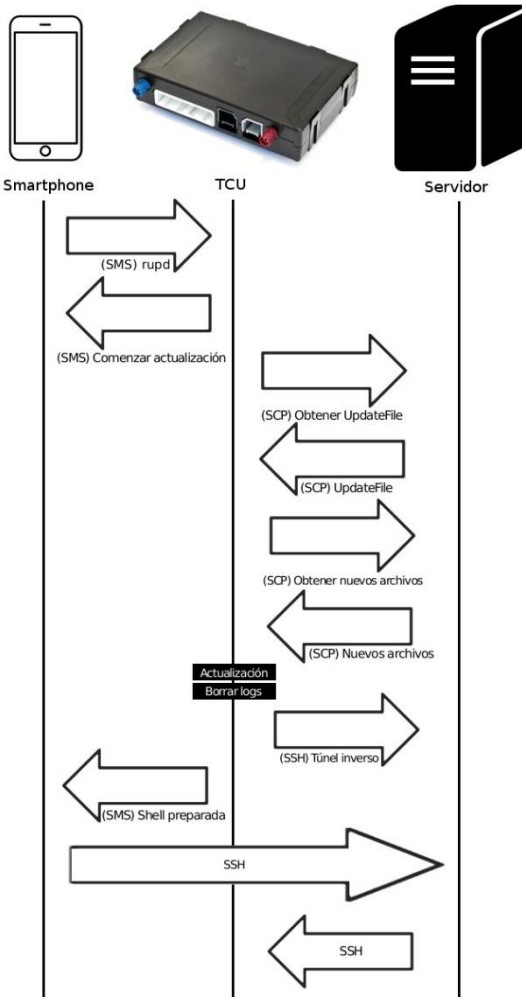


Fig. 11: Actualización desde un servidor suplantando la identidad con código malicioso.

El mayor fallo es que los archivos de actualización no están firmados criptográficamente, por lo que el servidor de actualización podría ser suplantado muy fácilmente. En la Universidad de California explotaron este ataque[19], de manera que fingiendo ser el servidor, incluyeron en el archivo de actualización un script que sustituía el firmware original. Éste, al ser ejecutado, inicia una *shell* con un túnel SSH y envía un SMS para informarnos de que el ataque se ha realizado con éxito. Una vez el atacante haya recibido el aviso con éxito, podemos conectarnos remotamente como *root*.

4.5 Acceso al servidor

Algunas unidades telemáticas empotradas en los vehículos contienen un servidor web al que se puede acceder mediante la dirección IP del dispositivo, ofreciendo varios servicios (hacer peticiones de manera remota para conocer la localización u otros parámetros, por ejemplo). Aunque la mayoría de proveedores de servicio usan NAT (Network Address Translation, permite asignar direcciones IP en redes privadas) para que cada vehículo no tenga una dirección individual, no siempre es así. En consecuencia, los servidores de cada coche podrían ser indexados por motores de búsqueda. Como detallamos en apartados anteriores, se han conocido vulnerabilidades de modelos que comparten las claves SSH en todos sus vehículos, por lo que se pueden buscar los servidores por su huella SSH (que debería ser única), dejando expuestas las IPs de miles de vehículos potencialmente vulnerables.

4.6 API

Algunos vehículos, como el Nissan LEAF (el coche eléctrico más vendido del mundo), están integrados con una aplicación móvil que permite la visualización de ciertas características. En este caso, por ejemplo, podemos comprobar a distancia el estado de la batería, encender o apagar el aire acondicionado, o ver cuándo fue la última vez que estuvo en marcha. Para ello hay una API que permite hacer estas consultas[23].

A Troy Hunt, director regional de Microsoft, le llamó la atención ver que en la petición que realizaba el vehículo al servidor no había datos autenticados de su unidad, sino que era una simple petición GET con unos códigos regionales, un *timestamp* y el VIN (*Vehicle Identification Number*, número de identificación del vehículo), el cual no es precisamente un secreto, ya que está expuesto en la parte posterior del vehículo. Esto quiere decir que se accede a la API anónimamente, devolviéndonos ésta un JSON con el estado actual del vehículo.

Al hacer consultas pasivas se puede observar el estado de cualquier vehículo simplemente conociendo su VIN, pero si se hace una consulta activa, que modifique algún parámetro del vehículo (encender el aire acondicionado, por ejemplo), en la respuesta JSON se puede observar que nos devuelve un ID del usuario y una clave de resultado.

Enviando una batería de peticiones al servidor variando los últimos cinco dígitos de los VIN se pudieron obtener datos de otros vehículos (algunos fallaban por no haber registrado sus datos en la aplicación o no coincidir su localización con el código geográfico). Este fallo se ha comprobado tanto en Noruega como Canadá y Reino Unido, y Nissan estuvo varios meses sin ponerle solución, antes de echar abajo el servicio.

Otros vehículos, como el BMW i3, tenían API disponible para el desarrollo, pero se cerró, por lo que algunos usuarios decidieron desarrollar su propia aplicación con comandos y peticiones de estado mediante ingeniería inversa, pudiendo abrir y cerrar puertas a distancia, así como encender y apagar las luces o la climatización[24].

5. Medidas de seguridad

Implementar medidas de seguridad en las redes de los vehículos es complicado por las restricciones que impone la naturaleza de las mismas. Se trabaja en un entorno controlado por tiempo, en el que los mensajes se tienen que transmitir de manera rápida y con cierta prioridad, ya que son sistemas críticos (una trama que indique que los frenos se han de activar no se puede permitir retrasos), además de sincronizar los ticks de reloj y las ranuras de tiempo para transmitir.

También tenemos que tener en cuenta que las ECUs tienen una capacidad de cómputo limitada, al encontrarnos en sistemas empotrados, además del gran coste que supondría para el fabricante dotarlas de procesadores lo suficientemente potentes como para llevar a cabo tareas que hasta recientemente no se habían considerado preocupantes. Por ejemplo, los microcontroladores S12XD, diseñados específicamente para automovilismo, tienen una CPU de 16 bits, EEPROM de 4 KB y una memoria flash interna de 512 KB. Es común encontrar memorias flash en lugar de RAM (que tendrían mayor velocidad) en las ECU por su coste más reducido. Los que se encargan de tareas más sencillas o de menor relevancia tienen CPUs de 8 bits, y en casos más extraños, de 32 bits, como el V850E1 usado en vehículos Toyota que se sospechó que tenían bugs de software que les hacían acelerar demasiado[25].

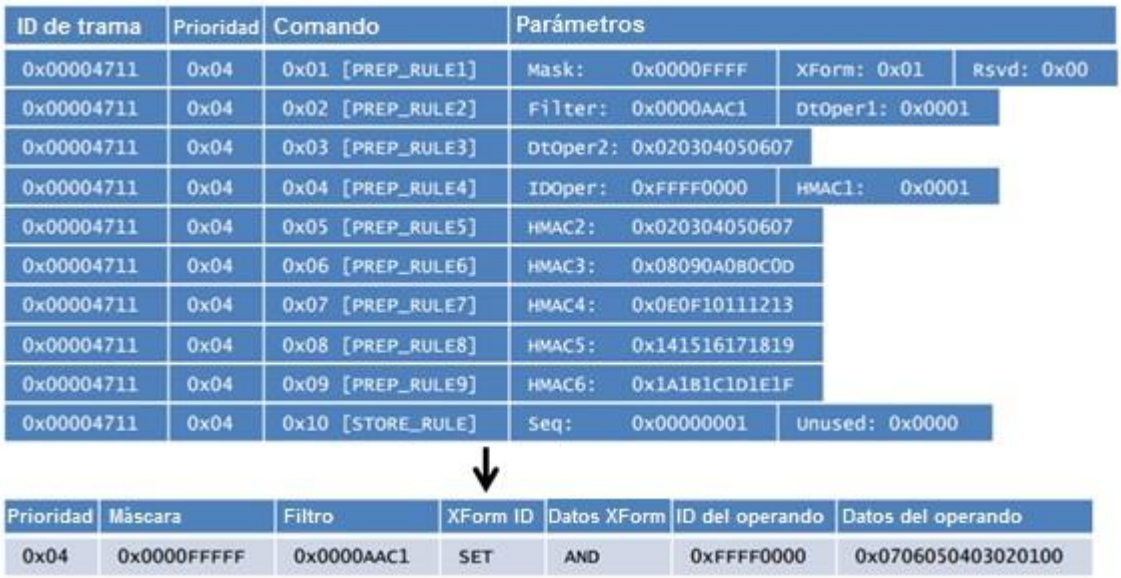
Juntando estas dos consideraciones, el tamaño de los paquetes no puede ser desmesurado: tienen que llegar a su destino y ser procesados en cierto tiempo. Esto hace que los códigos de redundancia o de autenticación no puedan ocupar demasiado espacio, puesto que ocuparían el ancho de banda que se debe dedicar a mensajes con importancia.

Aquí presentamos varias soluciones planteadas tanto por equipos de investigación de universidades como por empresas privadas que se dedican a este sector industrial, habiendo en varios casos distintas propuestas destinadas a resolver un mismo problema.

5.1 Firewall

Algunas empresas han desarrollado *firewall* para CAN con la intención de limitar el tráfico malicioso. Por ejemplo, Genivi cuenta con un dispositivo que se conecta a un servidor para tener las reglas que dejan paso o no a las tramas, evitando que las que no las cumplan entren por el módulo telemático[26]. Cada regla se firma con una clave específica del dispositivo y se actualizan *over-the-air*, pasándose como tramas CAN al *firewall*, que las valida y almacena.

Las reglas de este *firewall* tienen un método de validación con una máscara y un filtro: el ID se compara con la máscara y sólo los bits que queden a 1 se compararán con el filtro con un AND. Si pasan el filtro, serán interceptados. También cuentan con varios campos para aplicarles transformaciones, descartarlos o pasarlos sin modificación.



ID de trama	Prioridad	Comando	Parámetros			
0x00004711	0x04	0x01 [PREP_RULE1]	Mask:	0x0000FFFF	XForm: 0x01	Rsvd: 0x00
0x00004711	0x04	0x02 [PREP_RULE2]	Filter:	0x0000AAC1	DtOper1: 0x0001	
0x00004711	0x04	0x03 [PREP_RULE3]	DtOper2:	0x020304050607		
0x00004711	0x04	0x04 [PREP_RULE4]	IDOper:	0xFFFF0000	HMAC1: 0x0001	
0x00004711	0x04	0x05 [PREP_RULE5]	HMAC2:	0x020304050607		
0x00004711	0x04	0x06 [PREP_RULE6]	HMAC3:	0x08090A0B0C0D		
0x00004711	0x04	0x07 [PREP_RULE7]	HMAC4:	0x0E0F10111213		
0x00004711	0x04	0x08 [PREP_RULE8]	HMAC5:	0x141516171819		
0x00004711	0x04	0x09 [PREP_RULE9]	HMAC6:	0x1A1B1C1D1E1F		
0x00004711	0x04	0x10 [STORE_RULE]	Seq:	0x00000001	Unused: 0x0000	

Prioridad	Máscara	Filtro	XForm ID	Datos XForm	ID del operando	Datos del operando
0x04	0x0000FFFFF	0x0000AAC1	SET	AND	0xFFFF0000	0x0706050403020100

Fig. 12: Aplicación de las reglas del *firewall* de Genivi a una trama entrante.

Ya que hay varias reglas, se establece un orden de prioridad entre ellas, y se aplican en orden ascendente. Si no coincide con una, se pasa a la siguiente, y si se llega al final, se descarta la trama.

Los *firewall* también pueden tener integradas otras funcionalidades, tales como temporizadores y limitadores de tasa de envío para evitar ataques de denegación de servicio.

Sin embargo, se ha puesto en entredicho la utilidad de los *firewall* en este entorno debido a varias cuestiones. Los sistemas de comunicación en vehículos contienen una puerta de enlace (*gateway*) entre los distintos sistemas que los componen (como pueden ser MOST, FlexRay, LIN, y CAN dedicados a distintas funciones). Esto significa que cada uno de los controladores es capaz de enviar mensajes a cualquier otra ECU existente de cada una de las redes, y los datos que se transmiten abarcan todas ellas. Por ello, sin las medidas pertinentes, un solo bus vulnerable pone en peligro toda la red de comunicación. Esto, unido a la naturaleza *broadcast* de las redes, hace que si tenemos control de una unidad, el sistema completo esté comprometido[27]. La mayoría de profesionales del sector de seguridad en vehículos apuestan por técnicas de cifrado para asegurar la autenticidad e integridad de los mensajes enviados, así como un enfoque por capas a las distintas redes[28][29].

5.2 Cifrado y autenticación

Otra de las apuestas es cifrar los mensajes en los buses. Esto es especialmente dedicado porque la mayoría de protocolos funcionan de manera demasiado simple y añadir la carga de realizar el cifrado en unidades con tan poca capacidad de procesamiento como las ECU no es una tarea sencilla. Sin embargo, se están realizando avances en este aspecto.

Trillium ha desarrollado una tecnología llamada SecureCAN, diseñado para cifrar y administrar claves de *payloads* de menos de 8 bytes, a diferencia de otras tecnologías que necesitan bloques mayores[30]. Gracias a la posibilidad de cifrar tramas de este reducido tamaño se puede asegurar la confidencialidad de los mensajes de CAN y LIN en tiempo real. El cifrado simétrico permite cifrar, transmitir y descifrar con un umbral de un milisegundo, según la propia empresa. Sin embargo, señalan la necesidad de un *firewall* a pesar del cifrado.

El algoritmo TrilliumCipher combina transposición, sustitución y algoritmos de multiplexación de tiempo, incluyendo el *timestamp* dentro del propio texto cifrado, para evitar ataques de *replay*.

Cuando queremos autenticar los mensajes en CAN se plantean varias soluciones. La naturaleza *broadcast* dificulta esta tarea. Si se quiere implementar autenticación mediante MAC (*Message Authentication Code*) para asegurar además la integridad del mensaje cada nodo dentro de una subred ha de tener una clave distinta, puesto que si dos o más la comparten, cualquiera podría generar una MAC para cada mensaje fingiendo ser otra ECU.

Una solución a esto sería que cada uno de los nodos tuviera acordada una clave para cada posible receptor al que le llegase la trama enviada. Los MAC se pueden calcular de manera sencilla y en un tiempo corto, y así cada receptor podría estar seguro de que el mensaje procede del emisor que se identifica. El problema de ese planteamiento es que el tamaño se puede volver anormalmente grande al haber demasiadas ECUs en un sistema, además de tener los MAC una longitud demasiado grande (el payload de las tramas CAN es de 8 bytes y un MAC de tamaño completo podría ser cientos de veces mayor, en contraste con muchos mensajes CAN podrían ser simplemente *on* y *off* de un bit). Esto haría que las autenticaciones ocupasen la mayor parte del ancho de banda, lo cual es absurdo, además de probablemente tener que fragmentar la parte de autenticación al no caber en una sola trama y dar lugar a pérdidas o retransmisiones.

Para solventar esto se podría asignar a una MAC a un conjunto de mensajes, lo cual evitaría tener que enviar datos de gran longitud con tanta frecuencia. La problemática aquí radica en que si se perdiese una trama se tendrían que reenviar una gran cantidad de las mismas para poder autenticarlas correctamente.

Un equipo de la Carnegie Mellon University desarrolló un sistema para autenticación en sistemas empotrados dirigidos por tiempo. Se basa en aprovechar las retransmisiones periódicas de cada nodo (informando de su estado, valores de sus sensores…) para incluir en cada mensaje un trozo de MAC truncado y validándolo a lo largo de varios mensajes, con cierta tolerancia a fallos.

Para que uno de los nodos acepte un mensaje, debe recibir varios *consistentes*, es decir, que contengan los mismos datos o estén en un rango determinado, y todos ellos sean auténticos, usando funciones como SHA-256. Asume ciertas condiciones, como que no se instalarán o desinstalarán nodos sobre la marcha y que tienen la suficiente capacidad de cómputo como para calcular los MAC. Este sistema además, puede usarse en conjunto con protocolos que aseguren sincronización al nivel de milisegundos para evitar ataques de *replay*[31].

En el Fraunhofer ESK (Instituto para Sistemas Empotrados y Tecnologías de la Información) se optó también por el uso de MAC, pero de una manera distinta: aprovechando los campos CRC ya existentes de las tramas CAN para control de errores[32]. En este modelo, cada nodo tiene su propia clave secreta s_0 que hace de ser razonablemente larga (256 bits al menos), y se combina con unos bits que hacen de contador para evitar ataques de *replay*.

Concretamente, cada elemento del bus tendrá su propia clave secreta s_i generada a partir de variaciones de s_0, donde *i* es el contador de sesión antes mencionado. Este contador se concatena con la clave base s_0 y se le aplica la función de *hash* SHA-3, a la que llamaremos h_k, resultando en:

$$s_i = h_k(s_0|i)$$

Para no sobrecargar las tramas se recomienda truncar el resultado del *hash* a 256 bits. Según las consideraciones tomadas, en un vehículo que esté funcionando 8 horas al día, pasarían 20 años antes de que se produzca un overflow en *i* y se tuviera que cambiar s_0 en consecuencia, más de la esperanza de vida útil del vehículo.

Hay otros algoritmos que se han planteado como alternativa a HMAC (como GMAC, que tiene un gran rendimiento), pero que presentaban problemas como dificultad en generar MACs de poca longitud.

También hay opciones con criptografía asimétrica, a pesar de tener un mayor coste computacional. En Escript, un proveedor de soluciones de seguridad alemán, desarrollaron el siguiente esquema, basado en tener una “super-puerta de enlace” que conecte los distintos subsistemas (FlexRay, CAN, LIN, MOST…), de manera que cada grupo tenga un par de claves pública y privada y toda la interconexión de buses pase por ella[27]. A su vez, cada subsistema tiene una clave simétrica interna que comparten todos los nodos para poder cifrar sus

comunicaciones internas, así como la clave pública de la puerta de enlace, la cual contiene las claves para descifrar los mensajes de cada bus interno.

Se incluye también una firma digital para asegurar la integridad del mensaje, comprobándose la misma en la puerta de enlace, la cual puede iniciar actualizaciones de las claves.

Envío

1. C_1 = Cifrado(M, K_i) — Se encripta el mensaje M con la clave simétrica K_i.
2. S_M = Firma(C_1, SK_j) — Se firma C_1 con la clave secreta SK_j (opcional).
3. C = $C_1|S_M$ — Envía C compuesto de la concatenación de C_1 y S_M.

Recepción

1. M = Descifrado(M, K_i) — Descifrar C_1 al mensaje M con la clave K_i.
2. Verificar(S_M, PK_j) — Se verifica S_M con la clave pública PK_j (sólo en la puerta de enlace).
3. Objetivo $\in$ Aut_j — Se reenvía M a la subred objetivo si Aut_j lo permite (está autorizado, sólo en la puerta de enlace).

Fig. 13: Cifrado de los mensajes.

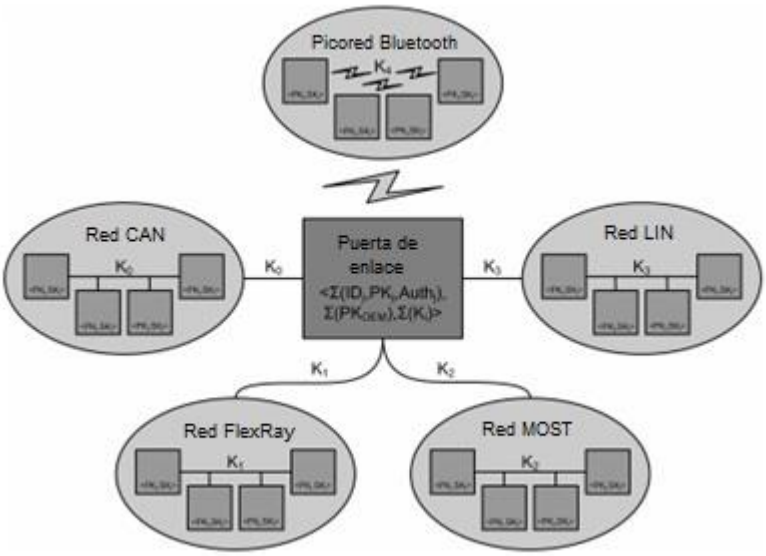


Fig. 14: Claves de sistemas y subsistemas según el esquema de Escript.

Por último, también debería haber forma de autenticar los servidores de actualización. Mientras que los archivos se transmiten mediante SSH en muchas ocasiones, no se verifica que la identidad del servidor sea fiable, pudiendo suplantarse fácilmente e instalar actualizaciones con código malicioso[19]. En los coches que permiten hacer esto mediante SMS se vuelve más complicado aún, ya que no es un protocolo que suporte una gran seguridad. Se pueden filtrar los números fiables mediante *whitelists* y *blacklists*, pero suplantar un número de teléfono no es una tarea complicada[33] (y muy práctica desde el punto de vista

del atacante de un vehículo) y estas listas no suelen ser de carácter privado, así que es un servicio que se aconseja desactivar.

5.3 Honeypots

Los *honeypots* son sistemas trampa que hacen creer al atacante que se encuentra en una red o sistema vulnerable para así poder monitorizar y analizar su actividad, así como estar al corriente de posibles nuevos ataques. Se pueden organizar en redes (llamados *honeynets*).

Se ha sugerido que los *honeypots* se conecten a la puerta de enlace de las conexiones remotas y simulen la red interna del vehículo, realizando una simulación realista. Teniendo en cuenta el coste, tamaño y capacidad de cómputo necesaria para llevar esto a cabo podría ser demasiada, ya que debería contar con su propio hardware y no afectar en absoluto al funcionamiento de la red real[34].

5.4 Análisis forense

Los diagnósticos actuales tan sólo incluyen análisis de errores, los cuales no se distinguen de las tramas enviadas maliciosamente. Sería muy útil mantener en un dispositivo similar a una caja negra un registro de todas las acciones que se han llevado a cabo, tanto las exitosas como no (fallos al transmitir en una ECU, retransmisiones, tramas perdidas, fallos de integridad, mensajes mal formateados…). Este sistema sería costoso porque necesitaría tener suficiente memoria y ser diseñada de manera resistente físicamente, además de estar protegida frente ataques, ya que si hay código malicioso que se infiltra en ella podría borrar, sobreescribir o modificar los datos allí guardados[35].

5.5 Aplicación de protocolos y estándares

Como hemos visto a lo largo de los distintos apartados, las unidades telemáticas de varios vehículos incorporan software que permite comunicarse por protocolos Telnet, transmitiéndose información en claro, sin ningún tipo de cifrado. Además fallos al aplicar las medidas de seguridad como compartir las claves SSH entre distintas unidades del mismo modelo son prácticas comunes. Esto nos lleva a la clara necesidad de desarrollar un ciclo de seguridad que aplicar al software de vehículos. La correcta implantación de unos protocolos y estándares podría evitar estos problemas en gran medida.

La cuestión que nos preocupa es que ni siquiera se aplican correctamente estándares ya existentes. En la especificación de CAN consta que las ECUs deben rechazar el comando “desactivar toda comunicación CAN” en situaciones que no sean seguras. En la práctica se ha demostrado que esto se puede hacer

incluso con el vehículo circulando a cierta velocidad[16]. Otras vulneraciones de estos estándares se dan al ver que es posible hacer *reflashing* de las ECU cuando el motor está arrancado o al observar cómo el desafío y la respuesta para autenticarse en la unidad telemática son *hardcoded* y ni siquiera hacen falta para instalar un nuevo *firmware*.

Además, en la mayoría de vehículos modernos hay un bus CAN de alta velocidad (para operaciones de mucha prioridad) y otro de baja velocidad, siendo implícitamente el primero más sensible y por tanto más custodiado. En consecuencia, en el estándar se detalla que las puertas de entrada entre los dos buses no podrán ser *reflasheadas* desde el bus de baja velocidad, para evitar que una ECU comprometida de tal nivel ponga en peligro el bus de alta prioridad[16]. Se ha comprobado empíricamente que hay vehículos que permiten que esto ocurra, poniendo en peligro la seguridad del vehículo.

5.6 Concienciación

Aunque los fabricantes admiten que actualmente el 100% de los vehículos en el mercado cuenta con tecnologías inalámbricas susceptibles a ataques remotos, virtualmente ninguno era consciente de ataques o problemas de seguridad recientes. Según un estudio conducido por el equipo del senador de Massachussets[7] (Edward J. Markey), de los 16 fabricantes que fueron tenidos en cuenta, tres no respondieron, doce afirmaron no tener conocimiento de tales incidentes, y tan sólo uno reportó un problema en el pasado, consistente en la retirada de la tienda de aplicaciones de una aplicación de Android que se comunicaba con el vehículo mediante Bluetooth, como medida preventiva.

Muchas compañías no supieron responder de manera concreta a qué medidas usaban para prevenir el acceso remoto (con ideas que van iban desde sistemas cerrados donde no se puede escribir código sin las herramientas adecuadas, hasta IDs únicos o dispositivos inalámbricos dedicados) y qué procesos usaban para asegurar los puntos de entrada (test de penetración, verificación de las entradas, virtualización…). Un punto común de todas ellas era que basaban su seguridad asumiendo que un atacante no podía entrar al sistema, sin saber cómo responder a una infiltración una vez se había realizado, excepto por una marca que detalló tener un sistema de *firewall* que vigilaba si había un comportamiento inusual en la red, pero sin entrar a ver el contenido de los mensajes. Se pudo ver cómo algunas apostaban por la *seguridad mediante oscuridad*, lo cual es especialmente controvertido cuando se trata de algoritmos que pueden tener debilidades de gran envergadura, como en este caso, ya que la vida de personas puede depender de ello.

Esto último nos lleva a la cuestión de la privacidad. El usuario medio no es consciente de que las compañías recaban una cantidad enorme de datos para uso privado, en parte porque el consentimiento informado es vago o

prácticamente inexistente, presentándose de distintos modos. Tomaremos como ejemplo el Renault Zoe, que presenta en los primeros arranques una pantalla de información con un texto haciéndonos saber que ciertas funcionalidades requieren recolección de datos, sin especificar cuáles, ni por parte de quién serán tratados o en qué condiciones, pudiendo aceptarlo o simplemente rechazarlo y no usar dicha funcionalidad.

S. Nurnberger, de Automotive Security, comprobó que los datos estaban siendo enviados a distintos servidores en varios países: los datos del navegador a TomTom y Atos en Holanda y Francia, respectivamente, y los de la unidad telemática a los servidores alemanes de Renault, a los cuales se podía acceder mediante una aplicación móvil[18]. Entre estos datos transferidos había algunos tales como el número de pasajeros en el vehículo en un momento (gracias a los sensores en los asientos), posición de GPS, alertas si se sale de una cierta zona geográfica, o si se conduce por encima de una velocidad determinada o después de una hora concreta.

Estos datos se pueden usar para estadísticas, asistencia o tareas más concretas (saber si alguien está usando el vehículo para transporte de pasajeros). Normalmente a los ojos del usuario medio, aceptar este acuerdo simplemente les permite poder usar el GPS o determinadas funcionalidades del ordenador a bordo que les hacen la conducción más sencilla. Por supuesto, este problema no atañe solamente a los vehículos, ya que es una falta de concienciación que vemos virtualmente en todas las aplicaciones, redes sociales y demás utilidades que requieran acceso a datos personales o sensibles.

6. Conclusiones

Como hemos podido ver en apartados anteriores, los diversos tipos de ataques que se pueden realizar a vehículos suponen riesgos en grados muy distintos. Desde una simple vulneración de la privacidad hasta peligro de muerte por manejar a distancia los mandos. Sin embargo hay ciertos aspectos de estos ataques que los convierte en más peligrosos aún.

Una característica a tener en cuenta de estos ataques es que se pueden borrar a sí mismos sin dejar rastro[16]. Se ha llegado a programar firmware malicioso que disparaba ciertas acciones en los mandos del vehículo para justo después *flashearse* a sí mismo, dejando el firmware original. Se ha intentado mantener un registro de este tipo de acciones en una especie de caja negra, sin embargo, ya hemos visto las dificultades que esto presenta en apartados anteriores.

También tenemos que comprender que no se trata de ataques informáticos de la clase con la que estamos acostumbrados a lidiar (virus, robo de información, corrupción de archivos…), sino que ponen en juego vidas humanas, incluso a gran escala, ya que hay varios tipos de ataques que son paralelizables (se podría comprometer a varios vehículos al mismo tiempo).

Asimismo, es muy importante enfatizar que el carácter remoto de los ataques juega un papel muy relevante, ya que ni siquiera se necesita acceso físico al vehículo o cercanía en el espacio para tomar el control de un vehículo.

A lo largo de estas páginas hemos observado cómo la seguridad en los vehículos no es un tema teórico sin traducción práctica, sino que se trata de un problema real que nos rodea. Hemos observado diversas noticias de actualidad en medios grandes, que van desde avisos sobre que se podría comprometer un vehículo con música desde un CD hasta informes sobre cómo se han detenido en seco coches en plena autovía de manera remota. La repercusión mediática de estos problemas (y otras cuestiones como la popularización de los coches sin conductor) ha llevado a los fabricantes a elaborar soluciones a los mismos, pero no es algo con lo que hayan estado acostumbrados a lidiar.

Por ejemplo, hacer una llamada desde los concesionarios a los usuarios para que lleven sus vehículos a una actualización (con la cual no van a notar ninguna mejoría en el funcionamiento) dejaría muchos vehículos con software obsoleto en circulación, así que otras empresas decidieron enviar *sticks* USB con las actualizaciones pertinentes, mientras que cada vez más apuestan por las actualizaciones *over-the-air*, aunque como hemos visto, estas últimas también tienen sus riesgos (autenticación de los servidores, actualización mientras el vehículo está en marcha…).

Como vemos, la mayoría de marcas no estaban concienciadas para esta problemática, y siguen sin estarlo en gran parte. Sus esfuerzos se centran en la prevención, pero no en cómo actuar frente a un ataque que ya se haya materializado, e ignoran incidentes ocurridos recientemente referentes a la seguridad en vehículos. Esto se ve apoyado por un desconocimiento por parte de los usuarios, que no son conscientes de la gran superficie atacable que supone un coche medio de los que se encuentren en el mercado, que tenga cientos de unidades de procesamiento en su interior y decenas de puntos de entrada susceptibles a ataques potenciales, además de sencillamente no estar al corriente de la situación o no darle importancia[36]. Sin embargo, las autoridades gubernamentales están tomándose el asunto con seriedad y mostrando advertencias al respecto[37] .

Por fortuna, esto parece estar cambiando. Al haber cada vez más novedades en el tema hay fabricantes que dedican esfuerzo a resolver estos problemas y empresas dedicadas especialmente a integrar soluciones que hacen frente a ataques[38], además de congresos a nivel internacional que abordan estos temas[39]. La industria en general también parece estar tomando un giro a mejor, ya que cada vez se usan y aplican más estándares (PassThru de implantación relativamente reciente, desde 2004 en Estados Unidos, al igual que CAN en Europa), y se idean ciclos de desarrollo de la seguridad para implantarla de manera correcta[40].

Estos esfuerzos podrían parecer desmesurados, pero hay que tener en cuenta las motivaciones detrás de los ataques. Hacer un sistema seguro, resistente a ataques y robusto, supone una gran inversión, que se vería reflejada en el precio y en el tiempo de desarrollo. Esforzarse tanto para, por ejemplo, un protocolo para abrir las puertas con una llave a distancia, cuando un ladrón simplemente podría romper una ventana, no merece la pena, por lo que hay que buscar un equilibrio entre el coste y la seguridad, sin olvidar que nos encontramos en un entorno empotrado, con una capacidad computacional reducida y sistemas en tiempo real cruciales. La realidad es que los problemas de seguridad más serios no se refieren a hurtos, sino a grupos de activismo *hacker* (en ocasiones apoyado por organizaciones gubernamentales), crimen organizado o exempleados que conocen fielmente el funcionamiento de los sistemas. No podemos olvidar que para llevar a cabo muchos de los ataques que hemos visto se necesitan conocimientos en profundidad de ingeniería inversa, protocolos de red y sobre todo, tiempo. Pero justo por ello, los efectos pueden ser devastadores.

Se ha analizado el vehículo de gama media en el mercado actual y sus posibles puntos de entrada. Hemos apreciado que además de puntos obvios como la conexión 3G o el GPS, hay puntos de entrada tan diversos como Bluetooth, los reproductores de música, los SMS, la apertura inalámbrica a distancia, o estándares industriales como el puerto de diagnóstico OBD-II.

Hemos visto los esfuerzos por implantar medidas de seguridad que resuelvan los problemas que se han ido encontrando en estos lugares, desde el esencial cifrado para asegurar confidencialidad y las técnicas para autenticación de los mensajes, hasta llegar a la conclusión de que hay sistemas que no aportan beneficios pero sí riesgos (actualizaciones vía SMS, inclusión de Telnet en el software…) y malas prácticas en la industria en general (no se cumple con los estándares, claves mal elegidas, falta de autenticación…). La aplicación de algoritmos y soluciones ya existentes en otros campos de la seguridad no es posible directamente, dadas las restricciones del ámbito en el que nos encontramos. No obstante, hemos podido ver cómo ciertos estándares (SHA-3, HMAC…) se usan de manera adaptada para poder realizar su función en los vehículos.

En general podemos observar cómo se está poniendo solución a un aspecto en el que nadie había pensado en la seguridad, y se está haciendo aprovechando estándares existentes, adaptando algoritmos, y elaborando protocolos para que los sigan distintas compañías. Se está empezando a tomar conciencia de que no sólo hay que hacer que las interfaces (como la unidad telemática con el mundo exterior) sean seguras, ni las puertas de entrada (*gateway* que conecte distintos buses, como CAN y LIN, por ejemplo), sino también las propias redes de manera interna, entre todos sus nodos, y dentro de las propias ECU asegurar su correcto funcionamiento, ya que pueden comprometerse y devolver resultados incorrectos[27].

7. Referencias

[1] - Intelligent & Connected - GM Has Long Envisioned a Day When Cars Don't Crash
http://media.gm.com/product/public/us/en/technology/home.detail.html/content/Pages/news/us/en/2014/Sep/0907-its-history.html

[2] – Wired - Hackers Remotely Kill a Jeep on the Highway—With Me in It
https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[3] – Network World - Thieves using a $17 power amplifier to break into cars with remote keyless systems
http://www.networkworld.com/article/2909589/microsoft-subnet/thieves-can-use-17-power-amplifier-to-break-into-cars-with-remote-keyless-systems.html

[4] – Strategy& - Connected Car Study 2015: Racing ahead with autonomous cars and digital innovation http://www.strategyand.pwc.com/reports/connected-car-2015-study

[5] – Business Insider - AT&T just took a big step to maintain its lead in wireless service for connected cars http://www.businessinsider.com/att-offering-unlimited-data-plans-for-connected-cars-2016-5

[6] – GSMA - 2025 Every Car Connected: Forecasting the Growth and Opportunity http://www.gsma.com/connectedliving/wp-content/uploads/2012/03/gsma2025everycarconnected.pdf

[7] – E. J. Markey, Tracking & Hacking: Security & Privacy Gaps Put American Drivers at Risk https://www.markey.senate.gov/imo/media/doc/2015-02-06_MarkeyReport-Tracking_Hacking_CarSecurity%202.pdf

[8] – Autoblog - ISIS could use a self-driving car to deliver a bomb
http://www.autoblog.com/2016/03/15/isis-terrorists-bomb-self-driving-cars-sxsw/

[9] – Sewell Direct - A Brief Explanation of CAN Bus
https://sewelldirect.com/learning-center/canbus-technology

[10] – H. Reuss, Extended Frame Format - A New Option of the CAN Protocol
ftp://lorca.act.uji.es/CAN/CAN%20bus%20specification.pdf

[11] – P. Richards, A CAN Physical Layer Discussion
http://ww1.microchip.com/downloads/en/AppNotes/00228a.pdf

[12] – S. Corrigan, Controller Area Network Physical Layer Requirements
http://www.ti.com/lit/an/slla270/slla270.pdf

[13] – M. Di Natale, Understanding and using the Controller Area Network
https://inst.eecs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf

[14] – BOSCH - CAN Specification (Version 2.0)
http://www.kvaser.com/software/7330130980914/V1/can2spec.pdf

[15] – S. Corrigan, Introduction to the Controller Area Network (CAN)
http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[16] – K. Koscher et al., Experimental Security Analysis of a Modern Automobile
http://www.autosec.org/pubs/cars-oakland2010.pdf

[17] – Cross Border Technologies - CAN Bus Firewall - cyber security for automobiles http://www.crossborder-technologies.com/node/85

[18] – S. Nurnberger, A Reverse Engineering Approach to Discover Privacy Issues in a Modern Car

[19] – I. Foster et al, Fast and Vulnerable: A Story of Telematic Failures
http://cseweb.ucsd.edu/~savage/papers/WOOT15.pdf

[20] – The Register - Stop the music! Booby-trapped song carjacked vehicles – security prof
http://www.theregister.co.uk/2016/01/26/hackers_can_take_full_control_of_car_os/

[21] – S. Checkoway et al, Comprehensive Experimental Analyses of Automotive Attack Surfaces http://www.autosec.org/pubs/cars-usenixsec2011.pdf

[22] – SiriusXM - Overview of Telematics-Based Emergency Assistance from SiriusXM Connected Vehicle Services
https://www.michigan.gov/documents/msp/SiriusXM_PSAP_Overview_Document_446655_7.pdf

[23] – Troy Hunt - Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs https://www.troyhunt.com/controlling-vehicle-features-of-nissan/

[24] – Terence Eden - Reverse Engineering the BMW i3 API
http://shkspr.mobi/blog/2015/11/reverse-engineering-the-bmw-i3-api/

[25] – Michael Barr, Unintended Acceleration and Other Embedded Software Bugs (Embedded Gurus) http://embeddedgurus.com/barr-code/2011/03/unintended-acceleration-and-other-embedded-software-bugs/

[26] – Genivi – CAN Bus Firewall
https://github.com/PDXostc/canfw/blob/master/doc/CAN%20Firewall.pdf

[27] – M. Wolf et al, Security in Automotive Bus Systems
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.728&rep=rep1&type=pdf

[28] – ComputerWorld - Firewalls can't protect today's connected cars
http://www.computerworld.com/article/2951878/telematics/firewalls-cant-protect-todays-connected-cars.html

[29] – T. van Roermund, Enabling the Secure Connected Car: A Multi-Layer Vehicle Security Framework (NXP Semiconductors)

[30] – EETimes - CAN Bus Can Be Encrypted, Says Trillium http://www.eetimes.com/document.asp?doc_id=1328081

[31] – S. Ganeriwal et al, Secure time synchronization service for sensor networks http://dl.acm.org/citation.cfm?id=1080809&dl=ACM&coll=DL&CFID=832867340&CFTOKEN=86218349

[32] – S. Bittl, Attack Potential and Efficient Security Enhancement of Automotive Bus Networks using Short MACs with Rapid Key Change (Fraunhofer ESK) http://publica.fraunhofer.de/eprints/urn_nbn_de_0011-n-2892491.pdf

[33] – A. Knight, Hacking the Connected Car: Leveraging the Communications Infrastructure and Electronic Control Units (ECUs) to Take Remote Control of a Connected Automobile (Brier & Thorn Inc.) https://alissaknight.com/2016/05/17/stories-from-the-garden-of-good-versus-evil-hacking-connected-cars/

[34] – P. Kleberger et al, Security Aspects of the In-Vehicle Network in the Connected Car http://www.syssec-project.eu/m/page-media/3/connectedcar-iv-2011.pdf

[35] – T. Hoppe et al, Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures http://www.sciencedirect.com/science/article/pii/S0951832010001602

[36] – Wired - Only One in 4 Americans Remembers Last Year's Epic Jeep Hack https://www.wired.com/2016/03/survey-finds-one-4-americans-remembers-jeep-hack/

[37] – Federal Bureau of Investigation (Public Service Announcement) – Motor Vehicles Increasingly Vulnerable To Remote Exploits https://www.ic3.gov/media/2016/160317.aspx

[38] – TowerSec – TCUShield http://tower-sec.com/tcushield/

[39] – CyberSecureCar http://www.cybersecurecar.com/

[40] – NCC Group - ASDL – The Automotive Secure Development Lifecycle https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2014/october/asdl-the-automotive-secure-development-lifecycle/