Détection d'intrusion pour des réseaux embarqués automobiles : une approche orientée langage

Ivan Studnia

Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *22 septembre 2015* **par :**

Ivan Studnia

Détection d'intrusion pour des réseaux embarqués automobiles : une approche orientée langage

---

JURY

Isabelle Chrisment	LORIA	Rapporteur
Ludovic Mé	Centrale-Supélec Rennes	Rapporteur
Mohamed Kaâniche	LAAS-CNRS	Examinateur
Philippe Quéré	Renault Guyancourt	Examinateur
Yves Roudier	Eurécom	Examinateur
Youssef Laarouchi	EDF	Directeur de thèse
Éric Alata	INSA Toulouse - LAAS-CNRS	Directeur de thèse
Vincent Nicomette	INSA Toulouse - LAAS-CNRS	Directeur de thèse

---

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

Laboratoire d'Architecture et d'Analyse des Systèmes

Directeur(s) de Thèse :

Vincent Nicomette, *Éric Alata* et *Youssef Laarouchi*

Rapporteurs :

Ludovic Mé et *Isabelle Chrisment*

Remerciements

Les travaux présentés dans ce manuscrit résultent d'une coopération entre le LAAS-CNRS à Toulouse et le Technocentre Renault à Guyancourt. Je remercie M. Jean Arlat, directeur du LAAS-CNRS de m'avoir accueilli au sein du laboratoire ainsi que M. Patrick Bastard et Mme Jacqueline Cucherat de m'avoir accueilli dans leurs directions respectives chez Renault.

Je remercie également les responsables successifs de l'équipe *Tolérance aux fautes et Sûreté de Fonctionnement informatique* (TSF) du LAAS-CNRS, Karama Kanoun et Mohamed Kaâniche, pour m'avoir permis de travailler au sein de leur équipe.

Je tiens à remercier tout particulièrement mes directeurs de thèse, Vincent Nicomette et Éric Alata ainsi que mon encadrant chez Renault, Youssef Laarouchi, pour la confiance qu'ils m'ont accordée, leurs nombreux conseils, leur bonne humeur et leur soutien sans faille. Ces remerciements s'étendent également à Mohamed Kaâniche pour toute son implication dans mes travaux. J'ai également une pensée émue pour Yves Deswarte, qui nous a quitté le 27 janvier 2014. Ce fut un honneur d'avoir pu travailler avec lui.

J'exprime toute ma gratitude à Mme Isabelle Chrisment et M. Ludovic Mé pour avoir accepté d'être les rapporteurs de mes travaux. Je remercie également l'ensemble des membres du jury pour avoir accepté de juger mon travail :

- Isabelle Chrisment, Professeur des Universités, LORIA
- Ludovic Mé, Professeur des Universités, Centrale-Supélec Rennes
- Mohamed Kaâniche, Directeur de Recherche, LAAS-CNRS
- Philippe Quéré, Chef de projet, Renault s.a.s.
- Yves Roudier, Maître de Conférences, Eurecom
- Youssef Laarouchi, Ingénieur, EDF R& D
- Éric Alata, Maître de Conférences, INSA Toulouse
- Vincent Nicomette, Professeur des Universités, INSA Toulouse

Je remercie Philippe Quéré pour m'avoir accueilli au sein de l'équipe *Logiciel Embarqué-Temps Réel* au Technocentre Renault. Je remercie également tous les membres de l'équipe : Youssef, Régis, Joris, Patrick, Pascal et Rémi pour leur accueil, leurs conseils et pour les bons moments que nous avons partagés. Toujours chez Renault, je tiens à remercier François Ougier, François Colet, Christophe Desfriches et Brigitte Lonc pour l'aide précieuse qu'ils m'ont apportée.

Je remercie chaleureusement tous les doctorants, post-doctorants, ex-doctorants, futur-ex-doctorants, ex-futur-doctorants et stagiaires de l'équipe TSF que j'ai pu rencontrer durant ces années et avec qui j'ai passé de très bons moments : Ludovic, Yann, Hélène, Pierre, Camille, Thibaut, Joris, Kalou, Amira, Quynh Anh, Benoît,

Roberto, Moussa, Thierry, Carla, Julien, William, Guillaume, Rui, Olivier, Anthony, Kossi, Maxime, Amina, Nam, Miruna, Fernand et ceux que j'ai oubliés (désolé).

Parmi ceux-ci, des remerciements particuliers sont adressés à Pierre André pour avoir toujours accepté de mettre à contribution ses nombreuses compétences techniques ainsi que sa bonne humeur pour aider un collègue dans le besoin. Je remercie aussi Yann Bachy, ses box et ses télés d'avoir investi et rénové un bureau (mention spéciale à la déco), puis de l'avoir partagé avec moi. Merci à Hélène Martorell pour m'avoir accompagné et grandement aidé lors cette expédition dans le grand Nord francilien. Merci à Ludovic Pintard pour son soutien dans les périodes difficiles, académiques comme cinématographiques.

Merci également aux doctorants rencontrés chez Renault – Sylvain Cotard, Antoine Blin et Rim Moalla – pour leur soutien.

Je remercie bien entendu aussi mes amis de longue date pour m'avoir supporté (et continuer de me supporter) de près ou de loin : Damien, Fabien, Valérian, Pierre-Antoine, Arnaud, Guillaume, Toto, Mathieu, Pierre-Yves, Tanguy, Madeleine, Romain, Vincent, Nikita, Anne-Céline et les autres.

Enfin, je tiens à remercier toute ma famille pour son soutien et ses encouragements constants. Un gros merci à mes parents et mes grands-parents qui ont contribué pour beaucoup à cette thèse. Pour conclure, tous mes encouragements vont au futur Docteur Vincent pour la suite de ses travaux.

« Coming back to where you started is not the same as never leaving. »

— Terry Pratchett

Table des matières

Introduction

Les véhicules modernes possèdent de nombreux mécanismes d'aide à la conduite qui permettent d'assister le conducteur (aides au freinage d'urgence, radars anti-collisions...), voire de prendre la main (régulateur de vitesse, parking automatique...) lors de l'exécution de certaines tâches. De telles fonctionnalités sont rendues possibles grâce à l'utilisation de calculateurs embarqués, appelés ECU (*Electronic Control Unit*, unité de contrôle électronique), qui contrôlent différents systèmes d'une voiture grâce à un ensemble de capteurs et d'actionneurs répartis dans le véhicule. Pour pouvoir coordonner leurs actions, ces ECU échangent des données entre eux via des bus de communication, le tout formant ainsi un véritable réseau embarqué. À terme, les évolutions de ces mécanismes permettront de concevoir des véhicules intégralement autonomes, capables de se déplacer et de s'insérer dans un trafic réel sans assistance humaine. Or, cet accroissement des fonctionnalités dévolues au logiciel entraîne une augmentation de la complexité de celui-ci. Cette complexité croissante des systèmes embarqués augmente ainsi le risque que des fautes apparaissent durant de leur conception. De telles fautes peuvent alors être à l'origine de défaillances dont les conséquences peuvent nuire gravement à l'intégrité du véhicule et à la sécurité de ses passagers. Afin de prévenir ce genre de situations, une norme [ISO11] créée pour l'automobile fournit des méthodes permettant de concevoir des systèmes électriques/électroniques (E/E) sûrs de fonctionnement. Cependant, les problématiques traitées par ces méthodes relèvent principalement de la sécurité-innocuité (*safety*) et ne tiennent pas compte de possibles interventions malveillantes sur les systèmes embarqués. En effet, historiquement, les systèmes embarqués d'une voiture constituaient un système fermé, c'est-à-dire que tous les éléments interagissent exclusivement entre eux et ne subissent pas d'intervention extérieure, sauf en des cas bien particuliers (passage chez un garagiste), ce qui limitait fortement les risques d'attaque informatique contre les calculateurs embarqués.

Or, en parallèle de leur évolution vers plus d'autonomie, les automobiles sont également devenues de plus en plus communicantes. Une voiture récente peut ainsi être équipée d'un port USB sur lequel connecter un support de stockage, être appairée avec des équipements mobiles via Bluetooth ou Wi-Fi ou encore posséder une connexion à Internet via les réseaux mobiles. Tous ces moyens de communication avec l'extérieur constituent alors autant de points d'entrée potentiels sur le réseau embarqué pour un attaquant. En cas d'intrusion réussie via un de ces points d'entrée, l'architecture réseau des automobiles actuelles fait qu'un attaquant pourra alors lire et émettre des messages sur les bus de communications afin de perturber le fonctionnement des ECU qui y sont rattachés. En fonction de l'étendue des capacités des ECU ciblés et de la criticité des systèmes qu'ils contrôlent, les conséquences de telles attaques peuvent aller d'une simple inconvenance pour les passagers jusqu'à une prise de contrôle, temporaire ou durable, du véhicule par l'attaquant. Ainsi, aux problématiques de sécurité-innocuité bien connues et maî-

trisées par les industriels viennent désormais s'ajouter des problématiques liées à la sécurité-immunité (*security*) des systèmes embarqués dans l'automobile qui ne peuvent plus être négligées. Pour être le plus efficace possible au sein du système complexe que représente une automobile moderne, une politique de sécurité doit être mise en place avec une vision d'ensemble du système à protéger. En effet, se contenter de moyens de sécurité préventifs laisse tout le réseau « interne » vulnérable si un attaquant parvient malgré tout à contourner ces protections. Ainsi, si une telle intrusion se produit, il est nécessaire de pouvoir identifier les actions de l'attaquant si celui-ci cherche à poursuivre son attaque sur le réseau interne. Venant en complément des systèmes de sécurité préventifs, la détection d'intrusion vise ainsi à détecter les violations de la politique de sécurité qui surviennent sur le système observé. Or, l'informatique embarquée dans une voiture n'est pas comparable à un ordinateur personnel et la conception de mécanismes de sécurité pour une automobile doit ainsi tenir compte de nombreuses contraintes. Par exemple, la durée de vie d'une voiture est d'environ vingt ans. Un mécanisme de sécurité conçu pour une automobile doit donc rester performant durant tout ce temps. De même l'architecture réseau d'un véhicule est spécifique à celui-ci : deux modèles différents, voire deux versions d'un même modèle, possèderont des calculateurs différents, des réseaux différemment agencés... Pour réduire les coûts liés à son déploiement à grande échelle, un mécanisme de sécurité doit donc être aisément portable d'une architecture à une autre. Enfin, les systèmes embarqués doivent fonctionner de manière autonome et transparente pour l'utilisateur (le conducteur). Celui-ci ne doit être mis à contribution qu'en cas d'urgence.

Cette thèse a été effectuée dans le cadre d'un partenariat CIFRE avec Renault S.A.S, et s'inscrit dans une démarche de mise en place de moyens de sécurité-immunité dans les réseaux automobiles. Elle concerne plus particulièrement la détection d'intrusion sur les réseaux embarqués en tenant compte des contraintes spécifiques à de tels systèmes.

Ce manuscrit s'agence de la façon suivante. Le premier chapitre est consacré à la présentation du contexte de nos travaux. Nous y présentons dans un premier temps les caractéristiques des systèmes embarqués dans l'automobile, les contraintes liées à leur développement, les tendances actuelles qui guident l'innovation puis nous décrivons les principaux protocoles réseaux utilisés dans les automobiles actuelles en nous focalisant principalement sur le plus employé d'entre eux : CAN. Dans une deuxième partie, nous détaillons les problématiques de sécurité informatique liées à ces systèmes puis concluons en présentant les orientations de nos travaux.

Dans le deuxième chapitre, nous décrivons les différents types d'attaques qui peuvent cibler les système embarqués d'une voiture et introduisons une classification de ces attaques selon quatre axes permettant de mettre en évidence la diversité des menaces existant contre les automobiles modernes.

Le troisième chapitre présente un état de l'art des mécanismes de sécurité, existants ou en cours de développement, conçus pour les réseaux automobiles embarqués. Le domaine de la sécurité-immunité appliquée à l'automobile est relativement récent et il n'existe pas encore de techniques de sécurisation standardisées pour

une voiture. Notre objectif a donc été ici de présenter et de catégoriser une grande diversité de moyens existants pour permettre d'élaborer une stratégie de défense la plus complète possible à l'avenir. Nous y décrivons notamment plus en détail le principe de la détection d'intrusion, qui constitue l'objet des chapitres suivants.

Le quatrième chapitre est consacré à la présentation de notre système de détection d'intrusion. En faisant l'hypothèse qu'un attaquant a réussi à obtenir un accès au réseau interne et est capable d'émettre des messages sur un ou plusieurs bus de communication de ce dernier, notre objectif est alors de déterminer, pour chaque message transitant sur un bus, si celui-ci est légitime ou non vis-à-vis des informations que l'on possède sur l'état de la voiture. Pour ce faire, notre système se base sur les spécifications du réseau embarqué (calculateurs et protocoles de communication) pour établir un langage régulier décrivant des séquences de messages interdites. La détection d'intrusion s'effectue alors en analysant les messages observés sur le réseau via des automates finis reconnaissant ce langage. À cet effet, nous présentons deux méthodes de détection basées sur deux automates différents permettant d'obtenir un meilleur compromis en fonction de la puissance de calcul et de la mémoire disponibles vis-à-vis de la complexité du système à surveiller.

Le cinquième chapitre présente une implémentation et une validation expérimentale des méthodes de détection décrites dans le chapitre 4. Pour ce faire, nous nous basons sur deux cas d'application simplifiés : un système de contrôle des feux et un régulateur de vitesse dont les différents éléments communiquent via un bus CAN.

Nous concluons finalement ce manuscrit en faisant une synthèse des travaux réalisés et en présentant quelques perspectives de travaux futurs.

Chapitre 1

Contexte des travaux et problématique

Sommaire

Les automobiles embarquent une quantité toujours plus importante de logiciel [Cha09] et un véhicule actuel embarque généralement entre 30 et 70 calculateurs (parfois bien plus [MV14]). De plus en plus de fonctionnalités d'une voiture sont ainsi prises en charge par du logiciel, celui-ci étant même responsable d'une grande majorité des innovations récentes dans le domaine [Bro05]. Ces calculateurs, appelés ECU (*Electronic Control Unit*, unité de contrôle électronique) contrôlent différents systèmes d'une voiture grâce à un ensemble de capteurs et d'actionneurs répartis dans le véhicule. Pour pouvoir coordonner leurs actions, les ECU ont besoin de communiquer entre eux, formant ainsi un véritable réseau embarqué. Comme tout autre système informatique, il n'est pas à exclure qu'un réseau automobile puisse

contenir des vulnérabilités, qu'un attaquant peut alors exploiter. Jusqu'à récemment, un réseau automobile embarqué pouvait être considéré comme un système fermé, sans lien avec l'extérieur, ce qui réduisait fortement les risques d'une attaque informatique. Or, les voitures modernes sont désormais équipées de moyens de communication variés tels que de l'USB, du Bluetooth, ou encore des réseaux mobiles (2,3,4G). Une voiture ne peut alors plus être considérée comme un système fermé, et les risques d'attaques informatiques dirigées contre leurs systèmes embarqués deviennent réels. Ainsi, la complexité accrue de ces systèmes couplée à la connectivité des automobiles modernes soulèvent des problématiques de sécurité informatique (au sens *security*, ou sécurité-immunité) auxquelles il devient urgent de répondre.

Dans ce chapitre, nous présentons les systèmes embarqués dans l'automobile ainsi que les problématiques de sécurité associées. Nous nous intéressons tout d'abord aux calculateurs embarqués, en nous concentrant plus particulièrement sur la partie logicielle. Ensuite, nous présentons les architectures réseaux déployées dans les automobiles, qui permettent à ces calculateurs de communiquer entre eux. Par la suite, nous définissons les concepts de sécurité informatique qui constituent la base de nos travaux puis nous attirons l'attention sur les problématiques de sécurité liées aux systèmes embarqués dans les automobiles. Enfin, nous concluons ce chapitre en présentant nos contributions dans le domaine de la sécurité des réseaux automobiles embarqués.

1.1 Systèmes embarqués dans l'automobile

Dans cette section, nous présentons dans les grandes lignes les caractéristiques des systèmes embarqués dans les automobiles. Nous énonçons tout d'abord les contraintes que ceux-ci doivent prendre en compte, puis nous présentons les problématiques liées à la conception d'un système logiciel embarqué dans une automobile ainsi que les solutions actuellement employées dans l'industrie. Enfin, nous nous intéressons aux tendances qui guident les innovations actuelles des systèmes automobiles embarqués.

1.1.1 Contraintes

Les ECU sont conçus, comme tout système embarqué, pour répondre à des besoins bien spécifiques. Lors de la conception de systèmes embarqués pour une automobile, les contraintes suivantes sont ainsi à prendre en compte.

Gestion de la qualité

L'automobile est un secteur extrêmement concurrentiel. Pour se démarquer parmi la grande quantité produite (plus de 67 millions de voitures en 2014 [1]), les constructeurs ont un besoin d'innovation constant qui entraîne l'arrivée régulière

1. `http://www.oica.net/wp-content/uploads//total-2014-Q4.pdf`

sur le marché de nouveaux modèles. Afin de rester compétitif dans un milieu hautement concurrentiel, la mise en place de politiques de gestion de la qualité (*Quality Management*) est essentielle. En effet, chercher simplement à proposer un produit de la meilleure qualité possible ne suffit pas si celui-ci est jugé trop cher par les clients ou si la durée du développement associé retarde trop la sortie du produit, faisant alors perdre l'avance sur les concurrents que pouvait apporter une innovation. Lors de la conception d'une automobile, l'enjeu pour un constructeur est ainsi de trouver le compromis idéal entre qualité, délai et coût. La part des systèmes électroniques (matériel et logiciel) embarqués dans le coût de conception d'une automobile est en hausse permanente, et représente aujourd'hui jusqu'à 40% du coût d'un véhicule haut-de-gamme [Cha09]. Par conséquent, ces systèmes embarqués font l'objet d'une forte attention de la part des constructeurs afin d'optimiser leur conception. À cet effet, la réutilisation de composants (matériels ou logiciels) développés lors de projets précédents (*carry-over*) ainsi que le recours aux composants sur étagère [2] sont des pratiques très répandues.

Contraintes physiques

Une automobile ne constitue pas un environnement stable et homogène. En fonction de leur emplacement, certains ECU peuvent alors être soumis à des contraintes physiques – telles que de fortes variations de température, de l'humidité ou encore des chocs – dans lesquelles ils doivent pouvoir continuer à fonctionner.

Durée de vie

La durée de vie d'une automobile est d'environ vingt ans, soit beaucoup plus qu'un ordinateur classique. La conception et le suivi des systèmes embarqués doivent donc tenir compte de cette durée de vie étendue. Il faut donc être capable d'assurer le suivi des systèmes embarqués durant toute cette période, aussi bien au niveau du matériel (il faut être capable de remplacer un calculateur défaillant) que du logiciel.

Ressources limitées

En conséquence des contraintes précédemment évoquées, la puissance de calcul et la mémoire d'un ECU sont généralement très limitées, comparées à un ordinateur (ou même un smartphone) actuel. Le logiciel embarqué doit donc être conçu pour optimiser les ressources dont il a besoin pour ne pas requérir de matériel plus performant, et donc plus cher.

Temps réel

Les systèmes embarqués doivent généralement se comporter en temps réel, c'est-à-dire qu'ils doivent être capables de fonctionner selon des contraintes temporelles

2. ou COTS, pour *Commercial Off The Shelf*. Ces composants, produits et utilisés en très grand nombre et dans des domaines potentiellement variés, ont ainsi l'avantage d'une maturité avancée pour un coût de production réduit.

précises. Certaines fonctions doivent ainsi s'exécuter dans un laps de temps donné afin de garantir la sûreté du véhicule et de ses passagers. On parlera de systèmes temps-réel stricts (*hard real time*) dans les cas où le non-respect de telles contraintes entraînerait des conséquences catastrophiques (c'est le cas des systèmes critiques d'une voiture, comme les freins par exemple). Par opposition, des systèmes dits temps-réel mous (*soft real time*) peuvent tolérer ces dépassements dans une certaine mesure (par exemple un service de *streaming* vidéo).

1.1.2 Architecture logicielle

Les constructeurs automobiles ne développent généralement pas eux-mêmes intégralement les systèmes embarqués dans leurs voitures. Les ECU peuvent ainsi provenir de différents fournisseurs, qu'ils aient été conçus spécifiquement en partenariat avec le constructeur ou qu'il s'agisse de COTS. Historiquement, le logiciel embarqué dans chaque calculateur d'une automobile faisait l'objet d'un développement spécifique (*ad hoc*). Or, avec l'augmentation continue de la complexité des systèmes embarqués dans les automobiles, les inconvénients d'un développement *ad hoc* (notamment une maintenance et une réutilisation très difficiles) deviennent de plus en plus coûteux et ne peuvent plus être négligés. Afin de gérer la complexité des systèmes embarqués développés aujourd'hui, l'industrie automobile se tourne vers une approche plus structurée du développement logiciel et a intégré des concepts tels que le *Model-Based Design*, ou conception basée sur les modèles [Mur10]. Plus généralement, le standard AUTOSAR (*AUTomotive Open System Architecture*), créé par les acteurs de l'industrie automobile, vise à permettre une cohérence globale du logiciel embarqué dans une automobile. Il est basé sur une architecture en couches afin de faire abstraction du matériel. Ainsi l'utilisation d'un standard facilite entre autres les réutilisations et les mises à jour [MFRV14] d'un logiciel précédemment développé mais également l'intégration des composants logiciels provenant de différents fournisseurs ou des composants sur étagères respectant ce standard au sein d'un système embarqué.

1.1.3 Tendances

Parmi les innovations récentes concernant les systèmes embarqués dans l'automobile, deux grandes tendances se dessinent : l'autonomie et la connectivité.

1.1.3.1 Autonomie

Les systèmes dits d'« aide à la conduite », assistent le conducteur de diverses manières et peuvent parfois contrôler partiellement le véhicule afin d'améliorer la conduite mais surtout la sécurité des passagers (et la sécurité routière en général). Les systèmes d'aide au freinage (ABS), les systèmes de géolocalisation (GPS) et les régulateurs de vitesse constituent des exemples très répandus de systèmes d'aide à la conduite. Dans la continuité de cette démarche, les ADAS (*Advanced Driver Assistance Systems*) proposent des fonctionnalités de plus en plus avancées, et permettent

FIGURE 1.1 – Un prototype de véhicule autonome de Google

d'automatiser de plus en plus de tâches liées à la conduite. Ainsi, certains véhicules modernes proposent des systèmes permettant au véhicule de rester dans sa voie de circulation, de ralentir et accélérer automatiquement dans les embouteillages ou encore de se garer automatiquement. À terme, l'objectif est donc une voiture entièrement autonome, c'est-à-dire capable d'effectuer l'intégralité d'un trajet, en présence d'autres véhicules qui eux ne sont pas nécessairement autonomes, sans intervention d'un humain. Si de tels véhicules ne sont pas encore commercialisés, de nombreux prototypes ont été dévoilés lors des années passées, le plus emblématique étant sans doute la voiture sans volant ni pédales présentée par Google en 2014 (voir figure 1.1).

Concrètement, cette tendance vers plus d'autonomie se traduit par une augmentation du contrôle opéré par les ECU sur la mécanique d'une voiture. Le contrôle de la colonne de direction ou des freins ne se fait ainsi plus hydrauliquement ni mécaniquement mais électroniquement. Dans de tels cas, on parle de contrôle numérique *by wire*, « par fil » (en fonction du système, on parlera ainsi de *steer-by-wire, brake-by-wire*, etc. que l'on regroupe sous l'appellation *X-by-wire*). Cette augmentation des fonctionnalités opérées par les ECU se traduit par un accroissement de la complexité des systèmes embarqués pour pouvoir appliquer ce niveau de contrôle.

1.1.3.2 Connectivité

Les automobiles modernes disposent de nombreux moyens de communication et sont capables d'échanger des données avec divers dispositifs. Ainsi, une voiture peut par exemple lire des morceaux de musique stockés sur une clé USB branchée au système multimédia, se connecter en Bluetooth avec un téléphone pour permettre de passer des appels « mains-libres » ou encore accéder à des services en ligne via les réseaux mobiles.

Cette connectivité va par ailleurs s'amplifier avec l'arrivée de mécanismes de coopérations entre véhicules, reposant sur des communications entre véhicules proches, dites V2V (*Vehicle to Vehicle*) et entre un véhicule et des équipements d'infrastruc-

tures fixes en bord de route, dites V2I (*Vehicle to Infrastructure*) [3]. Ces communications, permettront par exemple à un véhicule arrivant à un croisement de signaler sa présence et d'être par conséquent détecté malgré une mauvaise visibilité. De même, une voiture freinant brusquement alertera les automobiles qui la suivent afin que leurs conducteurs puissent réagir plus vite, ou que les véhicules eux-mêmes freinent automatiquement s'ils sont équipés des dispositifs d'aide à la conduite adéquats.

Ainsi, avec l'intégration de plus en plus poussée de moyens de communication dans les automobiles, la notion de réseaux automobiles englobe un ensemble bien plus large que le réseau de calculateurs embarqués. Comme illustré en figure 1.2, un véhicule est désormais à l'intersection de nombreux réseaux.

1. Tout d'abord, le réseau interne des calculateurs embarqués, (partiellement) accessible via la prise OBD (voir section 1.2.7 pour plus de détails).
2. Le véhicule est également interconnecté avec des équipements mobiles, tels un smartphone ou une tablette.
3. Le véhicule peut être connecté à Internet via les réseaux mobiles (2/3/4G) et accéder à différents services.
4. Au niveau des communications V2X, on distingue deux niveaux :
   (a) Localement, le véhicule communique avec d'autres automobiles.
   (b) Le véhicule communique également avec une infrastructure débarquée (bord de route) qui constitue elle-même un réseau à part entière.
5. Enfin, le véhicule reçoit des données par satellite (GPS).

Dans la section suivante, nous présentons les différents réseaux embarqués dans une automobile.

1.2 Les réseaux automobiles embarqués

Historiquement, les premières communications entre ECU se faisaient via des liaisons point-à-point. Avec l'augmentation du nombre de calculateurs embarqués dans une voiture, il n'a plus été envisageable de les faire communiquer ainsi. En effet, le câblage requis (de l'ordre de n^2 canaux de communication pour n ECU) serait alors trop volumineux, trop lourd et trop cher. Pour résoudre ces problèmes, les ECU sont connectés à un ou plusieurs bus où les messages émis sont multiplexés et diffusés à tous les autres calculateurs connectés. Différents protocoles (et supports physiques correspondants) de communication peuvent être employés en fonction des besoins en termes de débit et de fiabilité. Nous présentons ci-après les principaux types de réseaux utilisés pour les communications internes d'une automobile, en nous basant principalement sur la synthèse rédigée par Navet *et al.* [NSL09]. Nous nous intéressons ensuite à la structure des données effectivement échangées entre les ECU, lors du fonctionnement normal du véhicule et lors d'une session de diagnostic.

3. Ces deux aspects sont parfois regroupés sous l'appellation V2X.

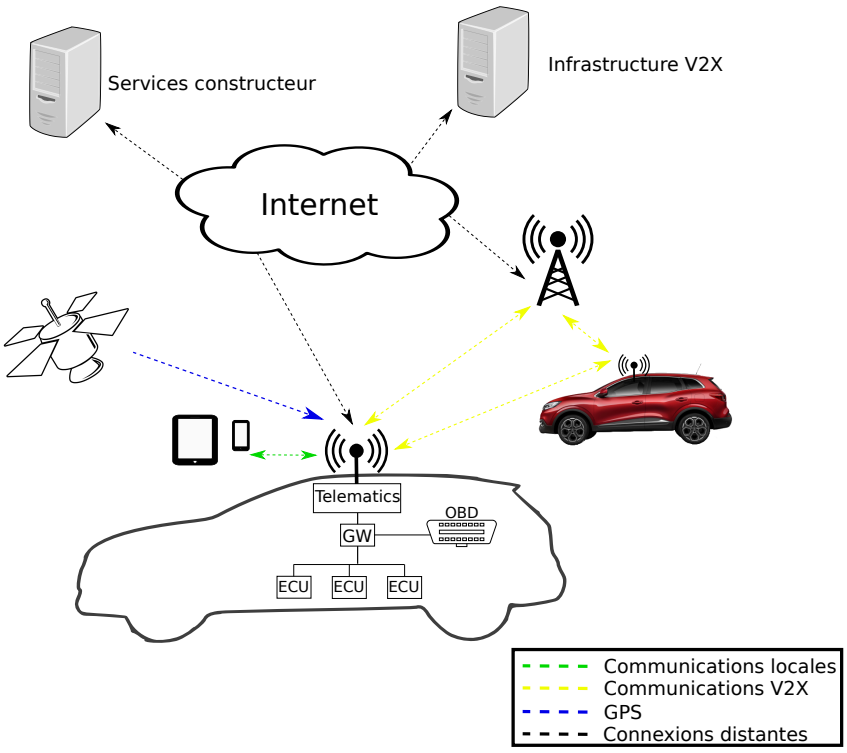
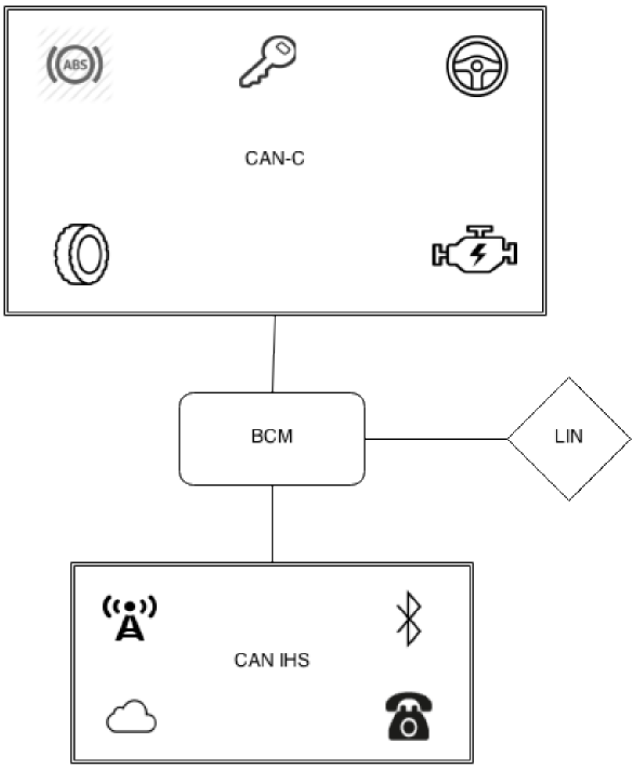


FIGURE 1.2 – Les différents réseaux d'une voiture moderne

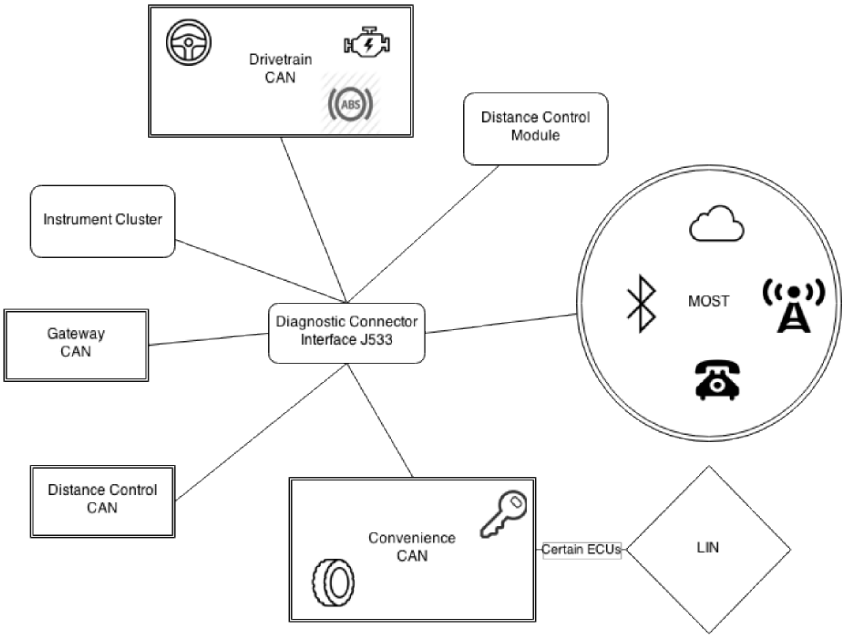
1.2.1 Architecture d'un réseau embarqué

Aujourd'hui, les réseaux automobiles offrent une grande diversité d'architectures : le nombre d'ECU, leurs fonctionnalités, leur disposition, les bus employés ou encore le contenu des messages peuvent varier fortement selon les constructeurs, les modèles, voire les séries. Cette diversité est très bien illustrée dans [MV14] où sont analysées (du point de vue de la sécurité informatique) 21 topologies de réseaux automobiles présents dans des modèles récents (produits entre 2006 et 2015). Nous reproduisons en figure 1.3 deux schémas tirés de ces travaux sur lesquels sont représentés les différents bus présents dans les véhicules, ainsi que la position de certains équipements : d'une part les outils de communication dont dispose la voiture, et donc les potentiels points d'entrée pour un attaquant, d'autre part la position des ECU responsables de fonctionnalités critiques (contrôle moteur, freinage, colonne de direction...). On voit ainsi que si les fonctionnalités proposées sont similaires, la disposition et l'interconnexion des équipements sur le réseau sont différentes.

Les fonctionnalités proposées par les systèmes embarqués d'une voiture sont très variées. En fonction de leurs objectifs, ces fonctionnalités ont des besoins différents en termes de performance, de sécurité ou de qualité de service. Un réseau automobile embarqué est ainsi généralement divisé en plusieurs *domaines* qui correspondent aux différentes familles de fonctionnalités. Le nombre et la définition de ces domaines peuvent varier. Cependant, le réseau interne d'une voiture moderne



(a) Chrysler 300 de 2014



(b) Audi A8 de 2014

FIGURE 1.3 – Deux représentations d'architectures réseaux de véhicules de 2014 présentées dans [MV14]

comprend généralement les domaines suivants :

- La motorisation (*powertrain*), qui regroupe les fonctionnalités relatives au contrôle du moteur et de l'injection.
- Le châssis, qui regroupe les fonctionnalités concernant la suspension, la direction et le freinage.
- L'habitacle (*body*), qui regroupe les fonctionnalités dites de confort.
- Un (ou plusieurs) domaine(s) dédié(s) à la gestion du multimédia, des IHM (Interfaces Homme-Machine) et des (télé)communications.

Ainsi, les architectures électroniques des automobiles modernes emploient différents types de réseaux, interconnectés par des ECU jouant le rôle de passerelles.

Les domaines précédemment évoqués se retrouvent par exemple dans l'architecture de référence du projet EVITA [HRS+09], présentée en figure 1.4 : chaque domaine possède un réseau dédié et chacun de ces réseaux est rattaché via un ECU passerelle à un réseau transverse.

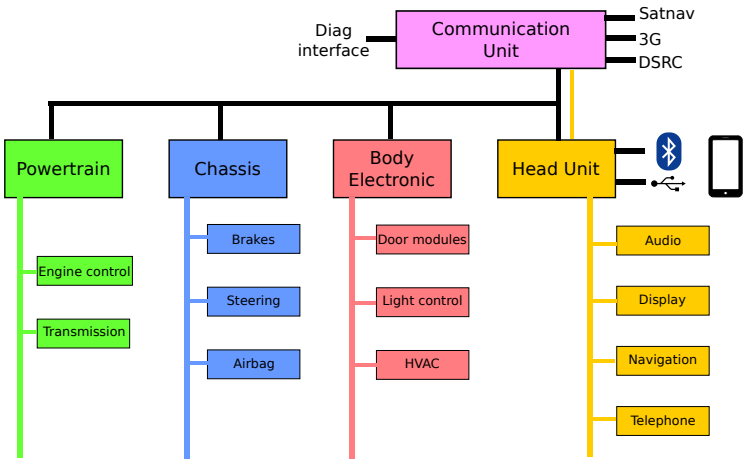


FIGURE 1.4 – Schéma de l'architecture de référence du projet EVITA

Historiquement, la SAE (*Society for Automotive Engineers*) a classé en 1994 les protocoles de communication en trois catégories (classes A à C) en fonction de leur débit et de leurs fonctionnalités sur le réseau. Les innovations arrivées par la suite, nécessitant des débits toujours plus élevés et de nouvelles fonctionnalités, ont provoqué l'ajout d'une quatrième catégorie (classe D). Cette classification est détaillée dans le tableau 1.1. Les réseaux de classe A proposent les débits de transfert les plus bas (moins de 10kb/s) et reposent sur des technologies à faible coût. Ils sont utilisés pour transmettre des données simples à moindre coût, typiquement entre un ECU et des capteurs ou actionneurs. Les réseaux de classe B (entre 10kb/s et 125Kb/s) sont conçus pour le transfert de données entre ECU lorsqu'il n'y a pas de contraintes de temps-réel fortes. Les réseaux de classe C (de 125Kb/s à 1Mb/s) permettent le

Classe	Débit	Usage	Exemples
A	<10kb/s	Contrôle de l'habitacle	LIN
B	10kb/s → 125kb/s	Transfert de données non-critiques	CAN-B (Low-speed CAN)
C	125kb/s → 1Mb/s	Communications temps-réel critiques	CAN-C (High-speed CAN)
D	>1Mb/s	Multimédia ou X-by-wire	MOST, FlexRay

TABLE 1.1 – Classification des réseaux automobiles selon la SAE

transfert de données pour des fonctions nécessitant des communications temps-réel (c'est le cas des domaines contrôlant moteur et châssis). Enfin, les réseaux de classe D englobent tous les réseaux dont le débit est supérieur à 1Mb/s. Ceux-ci sont employés aujourd'hui pour le transfert de données multimédia ou requérant une fiabilité très élevée, comme dans le cas des fonctionnalités *X-by-wire*.

Dans la suite de cette section, nous présentons différents exemples de réseaux automobiles embarqués, en nous arrêtant plus en détail sur CAN. Ce dernier étant actuellement le standard *de facto* pour les réseaux automobiles embarqués, il servira de point de référence lors de nos travaux.

1.2.2 CAN

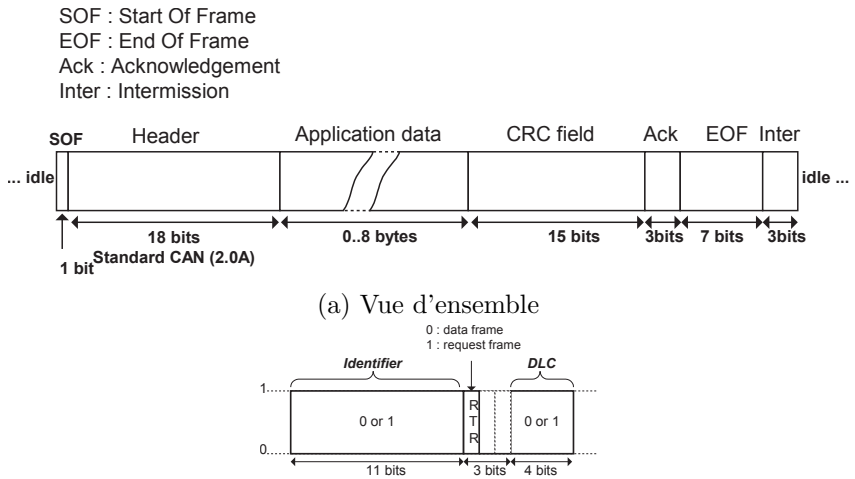
Initialement développé pour un usage automobile par Bosch en 1983, CAN (*Controller Area Network*) a été standardisé dans la norme ISO-11898 et est employé dans de nombreux autres domaines (comme l'aéronautique, par exemple). Il est aujourd'hui le standard *de facto* pour les communications embarquées dans une automobile. Principalement utilisé comme un réseau de classe C (généralement à 500kb/s), il est parfois également utilisé en classe B (à 125kb/s).

Il existe deux versions du protocole CAN, qui diffèrent par la taille de l'identifiant : CAN 2.0A (le CAN « standard ») avec un identifiant de 11 bits et CAN 2.0B (CAN étendu) avec un identifiant de 29 bits. Dans les automobiles, CAN 2.0A est majoritairement utilisé, puisque suffisant (il n'y a pas besoin de plus de 2^{11} identifiants). Par suite, nous désignons simplement par CAN le protocole CAN 2.0A. La couche physique du bus CAN utilise le « ET » logique : si un ECU émet un 0, le bus passe dans l'état 0, même si un autre ECU émettait un 1. Ainsi, 0 est la valeur dominante et 1 la valeur récessive. Les données peuvent être transmises de façon périodique, apériodique ou à la demande (dans une optique client-serveur). Ces données peuvent également être fragmentées sur plusieurs trames (la fragmentation éventuelle est gérée dans les couches supérieures).

Une trame CAN (figure 1.5a) se décompose comme suit :

- L'en-tête (*header*) (cf. figure 1.5b) contient l'identifiant de la trame, un bit RTR (*Remote Transmission Request*, qui définit une trame de données s'il est mis à 0 et une trame de requête de données s'il est mis à 1) et un champ DLC, pour *Data Length Code* qui indique la taille (en octets) des données transmises.
- Les données, pouvant occuper jusqu'à 8 octets.

– Un CRC (*Cyclic Redundancy Check*) de 15 bits qui permet de vérifier que les données n'ont pas subi d'altération accidentelle lors de leur transport.
– Le champ *Ack*, qui permet d'accuser réception de la trame. Cela permet à l'émetteur de savoir si son message a été reçu correctement par au moins un autre ECU (mais ne garantit pas que ce soit nécessairement le destinataire prévu initialement)
– Le champ EOF (*End Of Frame*, fin de trame) suivi d'une trame d'intermission, qui désigne le nombre minimum de bits à laisser passer avant l'émission d'un prochain message.



(a) Vue d'ensemble

(b) Détail de l'en-tête

FIGURE 1.5 – Structure d'une trame CAN standard

L'accès au bus se fait de manière asynchrone selon la technique CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*), c'est-à-dire que les communications se font selon les règles suivantes (cf. figure 1.6) :

– Il n'y a pas de tour de parole prédéterminé.
– Un nœud ne peut émettre que si le bus n'est pas actuellement occupé.
– Si plusieurs nœuds émettent simultanément, un arbitrage permet de déterminer lequel est prioritaire : c'est celui dont l'identifiant commence par le plus grand nombre de bits dominants, donc de 0. En d'autres termes, plus la valeur de son identifiant est petite, plus une trame est prioritaire. En cas d'égalité parfaite sur l'identifiant, le bit RTR est également pris en compte.

CAN possède plusieurs mécanismes de détection des erreurs. Nous mentionnons ici les trames d'erreurs (*error flags*) qui sont un type particulier de trame constituée de 6 bits dominants consécutifs. Lorsqu'un ECU détecte une erreur (par exemple lorsque le CRC de la trame reçue est incorrect), il émet une trame d'erreur pour avertir les autres membres du réseau. L'émetteur de la trame erronée doit alors tenter de la ré-envoyer lors de la prochaine étape d'arbitrage.

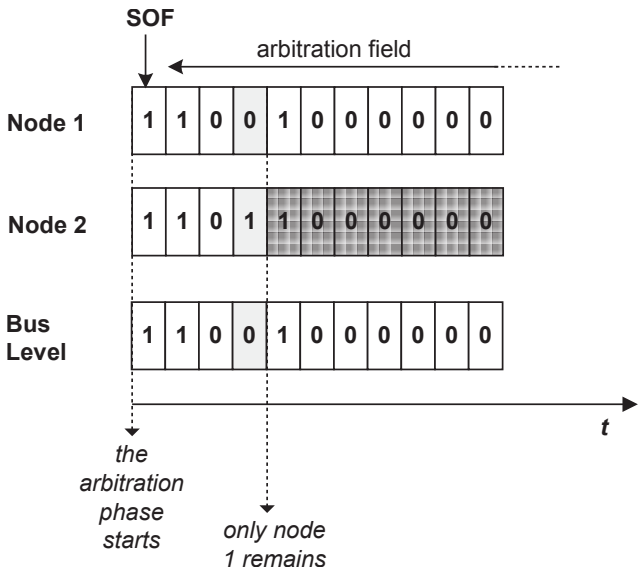


FIGURE 1.6 – Arbitrage entre deux trames émises simultanément sur CAN

1.2.3 LIN

LIN (*Local Interconnect Network*) est une solution à bas coût, proposant un faible débit (20kb/s maximum). LIN fonctionne selon un mode maître-esclave avec un nœud maître et jusqu'à 16 nœuds esclaves partageant un même bus. Le maître décide quelle trame doit être transmise en se basant sur une table de planification. Celle-ci contient la liste des trames qui doivent être transmises ainsi que les intervalles de transmission correspondants, et permet de garantir le déterminisme de l'ordre des transmissions. Lorsqu'une trame doit être transmise, le maître envoie un en-tête (qui peut être vue comme une requête de données) contenant un identifiant particulier (un réseau LIN peut posséder au plus 64 identifiants distincts) et l'esclave dont l'identifiant correspond envoie alors les données requises en réponse. Ce protocole a été conçu pour être utilisé dans les situations où les composants ne requièrent pas un débit élevé et peut ainsi permettre de réaliser des économies dans de tels cas. Il est généralement utilisé pour le contrôle des équipements de confort, tels que les vitres électriques

1.2.4 FlexRay

FlexRay est un protocole déterministe et tolérant aux fautes dont les premières spécifications ont été publiées en 2004 et aujourd'hui standardisé dans les normes ISO 17458-1 à 17458-5. Ce protocole possède deux canaux indépendants offrant chacun un débit allant jusqu'à 10Mb/s. Flexray définit des cycles de communication périodiques qui sont la concaténation de deux segments (voir figure 1.7) :

– Un segment statique où l'accès se fait de manière synchrone, selon un processus TDMA (pour *Time Division Multiple Acces* ou accès multiple à répartition dans le temps) : chaque trame se voit ainsi affecter un intervalle de temps

durant lequel elle devra être transmise.
– Un segment dynamique où l'accès se fait selon un mécanisme FTDMA (Flexible TDMA), qui permet de gérer des processus asynchrones, durant lequel les différents ECU se partagent de petits intervalles de transmission (*minislots*).

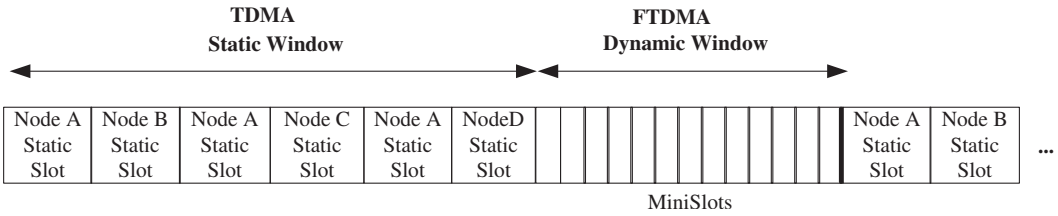


FIGURE 1.7 – Cycle de communication d'un réseau FlexRay comportant quatre éléments

Une trame Flexray contient jusqu'à 254 octets de données. FlexRay propose ainsi des débits plus élevés et des mécanismes de tolérance aux fautes plus importants que CAN, notamment grâce à son déterminisme plus fort mais aussi grâce aux deux canaux de communication permettant de faire de la redondance. Ces capacités étendues permettent un déploiement plus sûr de fonctionnalités temps-réel avancées, telles que les mécanismes relatifs au *X-by-wire*. Cependant, son coût encore élevé à l'heure actuelle ne lui permet pas de remplacer intégralement CAN dans les automobiles modernes.

1.2.5 MOST

MOST (*Media Oriented Systems Transport*) est utilisé pour transporter des données multimédia. Comme FlexRay, MOST possède un segment statique et un segment dynamique permettant respectivement une transmission synchrone et asynchrone des données, mais également un canal dit de contrôle permettant aux appareils connectés de demander l'attribution de canaux aux segments précédemment évoqués. Les canaux de transmission statiques sont typiquement utilisés pour le transfert de son ou de vidéo et les canaux dynamiques permettent par exemple de transférer des données téléchargées depuis Internet.

1.2.6 Évolutions futures

Les architectures réseaux évoluent en permanence d'un modèle à l'autre et, de la même manière que les réseaux embarqués se sont diversifiés pour répondre à différents besoins, de nouvelles possibilités sont envisagées pour le futur. Ainsi, Ethernet commence à faire son apparition dans les réseaux automobiles, que ce soit en remplacement de réseaux assurant le transfert de données critiques, comme CAN ou FlexRay [Bel11] ou pour le transfert de données multimédia.

Par ailleurs, afin d'alléger les véhicules du futur, [DVD11] envisage des réseaux automobiles futurs se passant de bus de communications, effectuant leurs transferts

de données via CPL (Courant Porteur en Ligne) ou grâce à des protocoles de communications sans fil. À l'heure actuelle, ces technologies ne sont pas assez matures pour concrétiser ces deux scénarios à l'échelle d'une voiture. Il existe cependant une fonctionnalité utilisant des communications sans-fil entre un ECU et ses capteurs : le système de mesure de la pression des pneus, ou TPMS (*Tire Pressure Monitoring System*).

1.2.7 Agencement des données

Les types de réseaux décrits précédemment ne définissent des règles que pour les couches basses de la pile protocolaire (à l'exception notable de MOST, qui implémente les 7 couches du modèle OSI), laissant l'implémentation des couches supérieures à la discrétion des constructeurs. Ainsi, en observant simplement le trafic réseau sur un bus, il n'y a pas a priori de moyen sûr de connaître la signification du contenu du champ de données (que nous appellerons désormais le **message**) d'une trame sans informations sur le contexte de son émission (quelle(s) action(s) l'a (ont) générée ou a (ont) résulté de sa transmission). Nous énonçons ci-après des principes généraux concernant le transfert de données via CAN.

Dans le cas de CAN, l'identifiant d'une trame détermine (en plus de sa priorité, cf 1.2.2) le type d'informations contenues dans les données. L'identifiant permet ainsi aux ECU connectés au bus de savoir si la trame contient un message leur étant destiné, et de connaître le sens de ce message. Nous pouvons néanmoins distinguer deux grandes catégories de messages : les messages « standards » et les messages de diagnostic.

Les messages standards désignent les messages que s'échangent les ECU lors du fonctionnement normal du véhicule. Ils peuvent contenir des informations sur l'état du véhicule qui pourront être utilisées par les ECU en ayant besoin (par exemple la vitesse actuelle) ou des instructions envoyées d'un ECU à un autre (« allumer les phares »). Nous désignons indifféremment ces instructions ou ces données en tant que **signaux**. Plusieurs signaux peuvent être inclus dans un même message. Si un message peut théoriquement être fragmenté et transmis sur plusieurs trames (via l'implémentation d'un protocole le permettant, comme ISO-TP, défini dans le standard ISO 15765-2), cela semble être peu utilisé en pratique. Le format de ces messages (taille, agencement du contenu) ne suit pas nécessairement de convention.

Les messages de diagnostic circulent sur le bus dans le cadre d'une réparation chez un garagiste lorsque celui-ci connecte un outil dédié (la « valise » de diagnostic) au véhicule. Concrètement, OBD (pour *On-Board Diagnostics*, diagnostic embarqué) désigne la capacité d'un véhicule à identifier et signaler les problèmes présents dans son architecture. Le port OBD (plus précisément un connecteur respectant le standard OBD-II ou une de ses variantes) est un connecteur standardisé (cf figure 1.8) et aujourd'hui obligatoire sur tous les véhicules vendus dans de nombreux pays (c'est notamment le cas aux États-Unis depuis 1996 ainsi qu'en Europe depuis 2001 pour tous les véhicules essence et 2004 pour les diesel). Le port OBD permet ainsi en s'y connectant d'initier une session de diagnostic avec les calculateurs du

FIGURE 1.8 – Le port OBD d'une Renault Fluence

véhicule afin de localiser et remédier rapidement aux dysfonctionnements qui se sont produits lors de l'utilisation de la voiture. Une session de diagnostic implique un dialogue entre les outils de diagnostic et les ECU concernés et des transmissions de données de taille importante (par exemple pour télécharger une nouvelle version du logiciel d'un ECU). Ainsi, les messages relatifs au diagnostic, plus complexes que les messages standards, suivent certaines conventions. De nombreux standards (ISO 14229, 14230, 15031, 15765) sont ainsi dédiés aux données échangées, et à la manière de les transmettre, lors d'une session de diagnostic. Là aussi, un constructeur peut choisir de dévier partiellement des standards. Aujourd'hui, il est possible d'acheter librement (ou de se fabriquer) un câble permettant de connecter un ordinateur au port OBD pour moins de 100€. En revanche, les logiciels de diagnostic conçus pour dialoguer avec le véhicule (et donc capables d'interpréter correctement les messages émis par les ECU du véhicule) ne sont généralement pas en vente libre.

Dans cette section, nous avons présenté les principales caractéristiques des réseaux automobiles. Avant d'aborder les problématiques liées à la sécurité informatique de ces réseaux, nous allons dans un premier temps définir les concepts de la sécurité informatique que nous manipulerons par la suite.

1.3 Terminologie de la sécurité informatique

Le terme de sécurité est ambigu. Il peut désigner aussi bien la sécurité-innocuité (*safety*) que la sécurité-immunité (*security*), qui sont deux concepts relevant de la sûreté de fonctionnement. Nos travaux s'inscrivant dans le cadre de la sécurité-immunité, cette section a pour objectif de définir le vocabulaire dont nous aurons besoin par la suite. Dans un premier temps, nous décrivons les concepts généraux de la sûreté de fonctionnement, introduits dans [LAD+96] et mis à jour dans [ALRL04] puis nous nous concentrons ensuite sur l'application de ces concepts au domaine de la sécurité-immunité.

1.3.1 Sûreté de fonctionnement

La sûreté de fonctionnement d'un système informatique est définie comme « la propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivre ». Le service délivré correspond au comportement du système perçu par ses utilisateurs. La sûreté de fonctionnement comporte trois axes principaux : les *attributs* qui la décrivent, les *entraves* qui empêchent sa réalisation et les *moyens* d'atteindre celle-ci (voir figure 1.9).

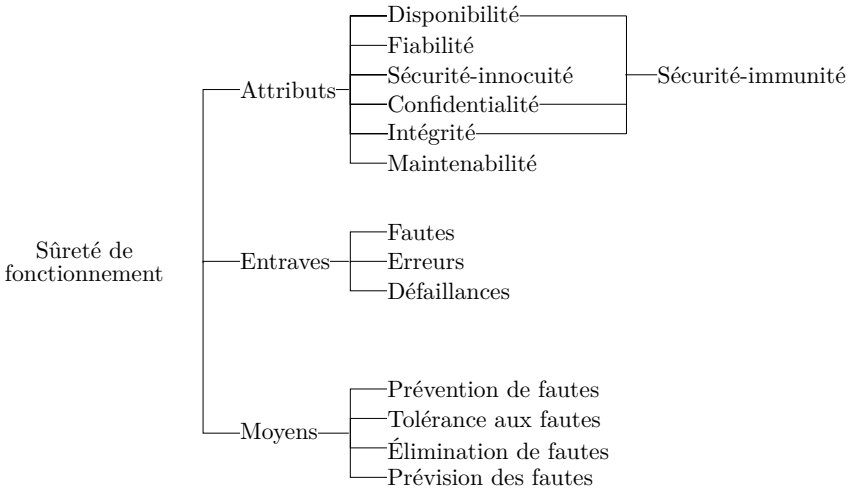


FIGURE 1.9 – Arbre de la sûreté de fonctionnement

La sûreté de fonctionnement peut être perçue selon les propriétés suivantes, appelées *attributs* de la sûreté de fonctionnement :

- Disponibilité : aptitude du système à être prêt à l'utilisation.
- Fiabilité : continuité du service.
- Sécurité-innocuité : absence de conséquences catastrophiques pour l'environnement.
- Confidentialité : absence de divulgations non autorisées de l'information.
- Intégrité : absence d'altérations inappropriées de l'information.
- Maintenabilité : aptitude aux réparations et aux évolutions.

Les attributs à considérer dépendent des applications auxquelles le système est destiné.

Ces attributs peuvent être mis à mal par des entraves. Une entrave est une circonstance indésirable, mais non-inattendue. Elle est la cause ou le résultat de la non-sûreté de fonctionnement. On distingue trois sortes d'entraves :

- Une *défaillance* survient lorsque le service délivré dévie de l'accomplissement de la fonction du système.
- Une *erreur* est la partie de l'état du système susceptible d'entraîner une défaillance.
- Une *faute* est la cause adjugée ou supposée d'une erreur.

Une faute est dite active lorsqu'elle produit une erreur. Par propagation, une erreur

crée de nouvelles erreurs. Une défaillance survient lorsque, par propagation, une erreur affecte le service délivré par le système. Cette défaillance peut alors apparaître comme une faute du point de vue d'un autre composant. L'enchaînement de ces entraves crée ainsi la chaîne fondamentale suivante :

$$\ldots \rightarrow défaillance \rightarrow faute \rightarrow erreur \rightarrow défaillance \rightarrow \ldots$$

Enfin, les moyens de la sûreté de fonctionnement permettent de minimiser l'impact des entraves sur les attributs. Ce sont les méthodes et techniques permettant de conforter les utilisateurs quant au bon accomplissement de la fonction du système. On distingue quatre catégories, en fonction de l'objectif visé :

- La prévention empêche l'occurrence ou l'introduction de fautes.
- La tolérance vise à fournir un service qui remplit la fonction du système en dépit des fautes.
- L'élimination réduit la présence (nombre, sévérité) des fautes.
- La prévision estime la présence, la création et la conséquence des fautes.

Nos travaux s'inscrivent dans le cadre de la **sécurité-immunité**, qui se définit comme une combinaison de disponibilité, confidentialité et intégrité. Le reste de cette section est dédié à la définition des concepts relatifs à la sécurité-immunité.

1.3.2 Sécurité-immunité

La sécurité-immunité vise à protéger un système contre les fautes malveillantes (ou *malveillances*) pouvant survenir lors de la conception ou de l'exploitation de celui-ci. Les concepts de sûreté de fonctionnement présentés précédemment sont génériques afin de pouvoir couvrir un grand nombre de concepts. Nous allons maintenant nous intéresser à leur application dans le contexte plus spécifique de la sécurité-immunité. Dans le reste de ce manuscrit, le terme *sécurité* lorsque nous employons l'expression *sécurité informatique* est à considérer au sens de sécurité-immunité, sauf si explicitement précisé.

1.3.2.1 Attributs

Considérés du point de vue de la sécurité-immunité, les attributs de la sûreté de fonctionnement que l'on cherche à garantir peuvent-être définis ainsi :

- **Confidentialité** : prévention de toute divulgation non-autorisée d'information.
- **Intégrité** : prévention de toute modification non autorisée d'information.
- **Disponibilité** prévention de toute rétention non-autorisée d'information.

Des propriétés supplémentaires, plus spécifiques, découlant des trois précédentes, peuvent également constituer des objectifs de sécurité-immunité. Les propriétés suivantes seront ainsi mentionnées dans ce manuscrit :

- **Imputabilité** : possibilité d'identifier le responsable d'une action donnée.
- **Non-répudiation** : irréfutabilité d'une action ayant eu lieu .

1.3.2.2 Malveillances

Les définitions qui suivent proviennent de la terminologie des malveillances introduite lors du projet MAFTIA [MAF03].

Il existe deux classes de fautes malveillantes : les logiques malignes et les intrusions. Une logique maligne est une partie du système conçue pour provoquer des dégâts (bombe logique) ou pour faciliter des intrusions futures (vulnérabilités créées volontairement). Elles peuvent être introduites dès la conception du système (par un concepteur malveillant) ou en phase opérationnelle (par l'installation d'un logiciel contenant un cheval de Troie ou par une intrusion). Une intrusion est définie grâce aux concepts d'attaque et de vulnérabilité.

- Une **attaque** est une faute d'interaction malveillante visant à violer une ou plusieurs propriétés de sécurité. C'est une faute externe créée avec l'intention de nuire, incluant les attaques lancées par des outils automatiques (vers, virus...).
- Une **vulnérabilité** est une faute, accidentelle ou intentionnelle, présente lors de la conception du système ou lors de son utilisation.
- Une **intrusion** est une faute malveillante correspondant au résultat d'une attaque qui a réussi à exploiter une vulnérabilité.

1.3.2.3 Moyens de lutte contre les malveillances

Les attaques, vulnérabilités et intrusions étant des fautes, les moyens de la sûreté de fonctionnement peuvent être appliqués pour améliorer la sécurité-immunité d'un système.

Prévention des fautes La prévention des fautes a lieu au début du cycle de vie du système et vise à empêcher l'occurrence ou l'introduction de fautes (donc dans le cas de la sécurité immunité, d'attaques, de vulnérabilités et d'intrusions). Concernant la sécurité immunité, les techniques relevant de la prévention des fautes incluent l'emploi de bonnes pratiques d'ingénierie ainsi que l'utilisation de politiques de sécurité.

Tolérance aux fautes La tolérance aux fautes est mise en œuvre en deux étapes : la *détection d'erreur*, qui permet de détecter un état erroné du système et le *rétablissement* dont le but est de ramener le système dans un état de confiance. Concernant les fautes malveillantes, un système tolérant aux intrusions est défini comme un système capable de s'auto-diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant une attaque [DBF91]. Parmi ces mécanismes, les méthodes de *détection d'intrusions*, dont il est question dans nos travaux, cherchent à détecter les atteintes à la politique de sécurité d'un système. La détection d'intrusion peut se faire selon deux approches : l'approche comportementale et l'approche par scénarios. La première se base sur une comparaison entre le comportement du système observé et une ré-

férence et détecte les déviations par rapport à cette référence. La seconde utilise une base de signatures de scénarios anormaux et détecte les comportements dont la signature est présente dans la base.

Parmi les autres mécanismes courants de la tolérance aux intrusions nous pouvons citer l'*authentification*, l'*autorisation* ou encore le *chiffrement*.

Élimination des fautes Il est difficile de développer correctement un système, ce qui fait qu'en dépit de l'application de techniques de prévention des fautes, certaines fautes peuvent persister, et peuvent alors être exploitées à des fins malveillantes. L'élimination des fautes cherche à en réduire le nombre, en s'appuyant sur des analyses statiques (revues, inspections), des vérifications formelles ou semi-formelles, des techniques de tests, etc.

Prévision des fautes Enfin, la prévision des fautes a pour but d'identifier les vulnérabilités, fautes, erreurs et modes de défaillances potentiels du système et d'analyser leur impact sur son comportement, vis-à-vis des propriétés attendues.

1.4 Problématiques de sécurité dans l'automobile

La complexité croissante des systèmes embarqués dans l'automobile entraîne mécaniquement une augmentation de la probabilité d'apparition de vulnérabilités dans les logiciels embarqués. Par ailleurs, la connectivité étendue des véhicules modernes provoque une multiplication des interfaces de communication entre le réseau interne et le monde extérieur. Ces interfaces constituent autant de points d'entrées potentiels sur le réseau pour un attaquant. La combinaison de ces deux tendances a soulevé brusquement un grand nombre de problématiques de sécurité (innocuité et immunité) à propos de la conception de ces systèmes embarqués.

1.4.1 Sécurité-innocuité

Les ECU peuvent être responsables de fonctionnalités critiques du véhicule. En fonction de leur criticité, de tels systèmes font l'objet de contraintes de sécurité-innocuité plus ou moins poussées afin de garantir leur bon fonctionnement. En effet, une défaillance du système multimédia, même si ennuyeuse pour les passagers, aura des conséquences bien moins dramatiques qu'une défaillance du système de freinage. Dans l'automobile, la norme ISO 26262 [ISO11] est la référence pour la sécurité-innocuité. Celle-ci définit notamment plusieurs niveaux de criticité, appelés ASIL (*Automotive Security Integrity Level*), pour chaque élément d'un système embarqué. Ces niveaux vont de A à D, A étant le moins critique et D le plus critique et sont déterminés suite à une analyse de risques. Pour chacun de ces niveaux, la norme propose différentes approches de conception possibles et de mécanismes à ajouter pour garantir un système sûr de fonctionnement.

1.4.2 Sécurité-immunité

Les motivations pour lancer une attaque contre les systèmes embarqués d'une automobile sont nombreuses. De plus, le véhicule, s'il est la cible de l'intrusion, peut ne pas être la cible finale de l'attaque, par exemple si l'objectif final est de nuire au propriétaire – ou au constructeur – d'une voiture.

Une analyse des risques liés aux incidents de sécurité-immunité dans les automobiles permet de définir une stratégie de défense plus efficace en fonction des besoins exprimés. À cet effet, [NPL08] propose par exemple une classification des ECU combinant les aspects *safety* et *security* via l'introduction des SEL (*Safety Effect Levels of security threats*), qui servent à évaluer l'impact d'un type d'attaque sur la *safety* du système embarqué. Cependant, si l'impact sur la sécurité-innocuité du véhicule est le plus important car des vies peuvent en dépendre, il ne représente pas la seule conséquence possible d'une attaque. En effet, les conséquences possibles d'une intrusion réussie varient en fonction des objectifs initiaux. Une intrusion peut certes déboucher sur une dégradation des aspects relatifs à la *safety* du véhicule, mais aussi provoquer une divulgation des données stockées dans les calculateurs (ces aspects ne sont pas mutuellement exclusifs). De plus, ces attaques peuvent également avoir un impact économique, pouvant concerner le propriétaire du véhicule comme le constructeur.

À l'heure actuelle, il n'existe pas encore de standard décrivant une classification équivalente aux ASIL (définis par l'ISO 26262) qui prenne en compte les incidents liés à la sécurité-immunité.

Attaques ciblant l'intégrité et la disponibilité des données

En ciblant l'intégrité et la disponibilité des données, c'est-à-dire en en ajoutant, en en modifiant ou en empêchant l'accès à celles-ci, un attaquant va chercher à perturber, voire à contrôler, le comportement d'un ou plusieurs systèmes du véhicule. De telles attaques peuvent alors impacter la sécurité-innocuité du véhicule, mais également causer des dommages économiques aussi bien au propriétaire du véhicule qu'au constructeur, suite à la dégradation de son image de marque.

Tout d'abord, un attaquant peut vouloir causer un dysfonctionnement du véhicule (ou d'un sous-système), par exemple car il souhaite prendre le contrôle de celui-ci ou provoquer une perte de contrôle de la part du conducteur. Les risques correspondants à de telles attaques sont équivalents à ceux envisagés dans le cadre de la sécurité-innocuité. En fonction de la criticité du système attaqué, les événements indésirables peuvent aller d'une inconvenance pour les passagers (par exemple si l'attaquant prend le contrôle du système multimédia) à des conséquences catastrophiques pour le véhicule, ses passagers ou son environnement (si l'attaque provoque une perte du contrôle du conducteur sur son véhicule).

Par ailleurs, de telles attaques peuvent également avoir un impact économique sur la cible. Du côté des usagers, en plus d'éventuelles réparations à effectuer suite à une attaque, il s'agira surtout des attaques dont l'objectif est de voler une voiture ou de la déverrouiller pour récupérer son contenu. Concernant les constructeurs,

les préoccupations grandissantes du public vis à vis des problématiques liées à la sécurité informatique font qu'une découverte de vulnérabilités dans les systèmes embarqués d'un de leurs véhicules pourrait entacher durablement leur image de marque. Une attaque réussie contre une automobile actuellement commercialisée, même aux conséquences mineures, peut impacter durablement les ventes du modèle concerné, voire du reste la gamme. Même sans preuve avérée permettant de relier un incident à une intrusion, fournir des réponses insuffisantes aux préoccupations du public [Mar15] risque de pénaliser les ventes d'un constructeur. Ainsi, une procédure judiciaire (*class action*)[4] a été intentée aux États-Unis par des consommateurs estimant que les constructeurs leur ont menti en ne mentionnant pas les risques de sécurité liés aux systèmes embarqués dans leurs automobiles.

Attaques ciblant la confidentialité des données

Un attaquant peut chercher à récupérer des informations contenues dans les systèmes embarqués. La notion de confidentialité des données couvre ici deux cas distincts.

D'une part, l'attaquant peut récupérer des informations confidentielles relatives au fonctionnement d'un calculateur, d'un sous-système ou du véhicule. On pensera par exemple à la récupération de code source propriétaire, qui pourrait permettre de fabriquer des calculateurs de contrefaçon.

D'autre part, l'attaquant peut récupérer des données relatives à la vie privée du conducteur (parcours GPS, carnet d'adresses ou toutes autres données synchronisées avec un smartphone par exemple) ou permettant d'identifier précisément celui-ci (comme les messages transmis par les capteurs de pression des pneus pouvant permettre de traquer une voiture à distance [RMM+10]). L'évaluation de la gravité de tels scénarios dépend des contraintes concernant la le respect de la vie privée des utilisateurs que se fixent les constructeurs.

1.4.3 Contraintes liées à la sécurité-immunité

En plus des contraintes liées au développement de systèmes embarqués évoquées en 1.1.1, la conception de mécanismes de sécurité pour les systèmes automobiles embarqués doit idéalement prendre en compte deux contraintes supplémentaires : la compatibilité et l'autonomie.

Compatibilité

La contrainte de compatibilité couvre en fait deux aspects : la rétrocompatibilité et l'interopérabilité.

Concernant la rétrocompatibilité, un mécanisme de sécurité conçu pour une automobile doit idéalement pouvoir être compatible avec les technologies présentes

4. `http://www.networkworld.com/article/2895535/microsoft-subnet/ford-gm-and-toyota-are-being-sued-for-dangerous-defects-in-their-hackable-cars.html`

dans les véhicules actuels. En effet, un mécanisme de sécurité ne requérant pas de fortes modifications d'une architecture existante permet de réduire les coûts liés à sa mise en place. Il en va de même lors de la conception d'un nouveau modèle de voiture, pour lequel il est intéressant de pouvoir réutiliser des composants développés lors de projets précédents.

L'interopérabilité désigne le fait que deux systèmes de sécurité remplissant une fonction similaire puissent coopérer. Par exemple, dans le cadre des communications V2X, il est important que des véhicules de marques différentes puissent communiquer entre eux et n'en soient pas empêchés parce qu'ils n'utilisent pas les mêmes protocoles de chiffrement.

Autonomie

Dans une voiture, il est important que le conducteur puisse concentrer toute son attention sur la conduite en elle-même. Par conséquent, les mécanismes de sécurité doivent être le plus autonomes possible et ne requérir l'attention du conducteur que lorsque la situation l'exige réellement.

1.5 Conclusion : Objectifs de la thèse

Dans ce chapitre, nous avons introduit le contexte général de nos travaux. Nous avons tout d'abord présenté les systèmes embarqués dans les automobiles, les contraintes liées à leur conception ainsi que les tendances majeures qui guident actuellement les innovations dans ce domaine. Ensuite, nous avons décrit les réseaux embarqués qui permettent de connecter entre eux les ECU que contient une automobile. Nous avons ensuite présenté les concepts fondamentaux de la sécurité informatique et attiré l'attention sur les implications potentielles d'une sécurité insuffisante dans les réseaux automobiles embarqués. Il apparaît ainsi que, suite à l'augmentation continue de la complexité des systèmes automobiles embarqués, aux capacités de contrôle toujours plus étendues des ECU et à une connectivité croissante, les problématiques de sécurité-immunité liées aux systèmes embarqués dans les automobiles ont pris une importance croissante et ne peuvent plus être négligées. Nos travaux portent sur la mise en place de moyens de sécurité-immunité dans les réseaux automobiles, et concernent plus précisément la détection d'erreurs au sein de ces réseaux. Au regard de la durée de vie d'une voiture et de la fréquence des mises a jour, il est illusoire, pour traiter les attaques, de se contenter de celles connues au moment du développement de la voiture. Par conséquent, un effort doit être mené pour réaliser une détection d'erreur efficace, même vis-a-vis des attaques qui seront découvertes dans le futur. De plus, se prémunir contre une grande diversité de menaces, une politique de sécurité ne doit pas reposer intégralement sur un seul type de moyen de protection mais utiliser plusieurs mécanismes agissant à divers niveaux des systèmes à protéger. Ainsi, pour concevoir notre système, nous avons tout d'abord cherché à caractériser d'une part les différentes menaces pouvant cibler des automobiles modernes et d'autre part les solutions de sécurité-immunité

conçues pour les automobiles existant actuellement. Notre manuscrit s'agence autour des thèmes suivants :

- Les besoins en termes de sécurité-immunité des automobiles sont encore relativement récents, et relativement peu de données sont disponibles sur le sujet. Il n'existe ainsi pas, à l'heure actuelle, de standard décrivant l'application de politiques de sécurité dans l'automobile, ni de certifications correspondantes. Concernant les menaces, plusieurs publications ou preuves de concepts se concentrant sur certaines menaces en particulier sont apparues dans les dernières années. Cependant, à notre connaissance, aucune classification officielle n'existe. La première partie de nos travaux a donc consisté en une prise de connaissance des spécificités du monde de l'automobile vis-à-vis de la sécurité informatique [SNA+13b]. Nous avons ainsi établi une classification des menaces pouvant affecter les systèmes embarqués dans les automobiles, que nous présentons dans le chapitre 2.
- Par ailleurs, il n'existe pas encore de standards concernant le déploiement de moyens de sécurité-immunité pour les systèmes embarqués dans le domaine automobile. Ainsi, le chapitre 3 est consacré à un état de l'art du domaine de la sécurité-immunité pour les systèmes embarqués dans une automobile. Nous y présentons notamment les projets européens ayant trait à la sécurité-immunité dans les automobiles et introduisons la notion de défense en profondeur. Celle-ci nous permet d'illustrer la nécessité d'employer des mécanismes de défense complémentaires afin de protéger au mieux un système complexe tel qu'une automobile moderne contre les intrusions. Nous y présentons enfin les différents types de mécanismes de défense adaptés à l'automobile présents dans la littérature.
- Comme nous l'avons vu dans ce chapitre, il existe une très grande diversité d'implémentations des systèmes embarqués dans les automobiles : chaque modèle de voiture possède ses spécificités. Pour pouvoir être adopté par l'industrie, un mécanisme de défense doit donc pouvoir s'adapter relativement aisément à ces diverses architectures. Les chapitres 4 et 5 de ce manuscrit concernent ainsi la conception et l'évaluation d'un système de détection d'intrusions pour les réseaux automobiles embarqués qui ne nécessite pas de modifier le système à surveiller [SNA+13a, SAN+15]. Notre objectif est ainsi d'obtenir une solution pouvant s'adapter aisément sur diverses architectures, existantes ou futures.

Chapitre 2

Attaques ciblant les architectures automobiles embarquées

Sommaire

Dans ce chapitre, nous analysons en détail les différents types d'attaques susceptibles de cibler les calculateurs embarqués dans une automobile moderne. Nous nous intéressons ainsi à tous les éléments prenant part à de telles attaques pouvant permettre de classifier les actions d'un attaquant.

Le chapitre s'agence comme suit. Dans la section 2.1, nous nous intéressons aux différentes étapes d'une attaque, depuis les motivations de l'attaquant jusqu'aux conséquences de l'attaque, souhaitées ou non, qui peuvent découler de leurs actions.

Dans la section 2.2, nous présentons et catégorisons les attaques que nous qualifierons d'internes, c'est à dire des attaques réalisables contre le réseau embarqué d'une voiture lorsqu'on est directement connecté à celui-ci.

La section 2.3 se concentre sur la description et la classification des attaques lancées à distance contre le véhicule, illustrées lorsqu'il y en a par des exemples tirés de la littérature.

Finalement, dans la section 2.4, nous faisons la synthèse des différents scénarios d'attaques présentés dans les sections précédentes en proposant une caractérisation de ces attaques selon quatre axes principaux.

2.1 Décomposition d'une attaque

Les attaques contre les équipements embarqués par une automobile peuvent prendre plusieurs formes. Il convient de distinguer :

1. Les attaques visant à altérer le comportement normal du véhicule sans pour autant pénétrer son réseau, par exemple en émettant un signal pour perturber un radar anti-collision. Ces attaques visent principalement les interfaces du véhicule.
2. Les attaques informatiques, dont le but est de prendre le contrôle de tout ou d'une partie du réseau embarqué dans le véhicule grâce à l'exécution par des calculateurs du véhicule de commandes envoyées par l'attaquant, via du code injecté ou des trames réseau malveillantes émises. Ces commandes sont envoyées à un point d'entrée tel qu'une prise usb, le lecteur cd, la prise de diagnostic, etc. Deux sous catégories peuvent être distinguées :
   (a) Les attaques informatiques où l'attaquant ne cherche pas spécialement à accéder au réseau embarqué mais est intéressé par les données qu'il peut récupérer suite à la compromission d'un élément du réseau embarqué (qui est par conséquent également la cible finale). Une attaque menée contre le système télématique afin d'écouter les conversations ayant lieu dans une voiture entre dans cette catégorie.
   (b) Les attaques informatiques où l'attaquant va chercher à interagir avec le reste du véhicule depuis son point d'entrée via le réseau embarqué.

Lors de nos recherches, nous nous sommes spécifiquement intéressés aux attaques informatiques, et plus particulièrement aux attaques se propageant via le réseau embarqué. Nous ne considérons pas pour autant que les autres types d'attaques sont négligeables. Ceux-ci constituent également des sujets d'étude intéressants mais sont hors du cadre de nos travaux.

2.1.1 Objectifs de l'attaquant

Avant de s'intéresser aux scénarios d'attaque potentiels, il faut tout d'abord se demander quels sont les intérêts pouvant motiver une attaque informatique contre le système embarqué dans un véhicule.

Vol

La motivation au premier abord la plus évidente est peut être le vol. Plusieurs possibilités sont envisageables pour voler un véhicule (ou son contenu) en s'appuyant sur l'électronique qu'il embarque. Tout d'abord, une attaque réussie contre

le système d'ouverture à distance peut permettre de déverrouiller la voiture, voire de la démarrer dans le cas de véhicules ne nécessitant plus un contact pour démarrer (PKES *Passive Keyless Entry and Start*). Une autre possibilité est d'utiliser une vulnérabilité d'un protocole de communication sans fil pour prendre le contrôle du calculateur correspondant puis de s'en servir pour déverrouiller discrètement la voiture et désactiver ses mécanismes d'antidémarrage ou une éventuelle alarme en envoyant des instructions sur le bus. Dans de tels cas, le gain de l'attaquant correspond à la valeur du véhicule volé ou celle des biens qu'il contient.

Modification des capacités du véhicule

Nous regroupons dans cette catégorie l'ensemble des cas où le propriétaire du véhicule en est également « l'attaquant ». Celui-ci va alors chercher à effectuer des modifications non autorisées sur le code ou les données contenus dans un ECU. Par exemple, il peut vouloir en diminuer le kilométrage pour le revendre plus cher, altérer le fonctionnement du moteur pour tenter d'obtenir plus de puissance ou encore installer de nouveaux programmes illégalement obtenus sur son ordinateur de bord. Un attaquant peut également tenter de contourner des mécanismes d'authentification afin d'installer des ECUs différents de ceux approuvés par le constructeur afin de réaliser des économies. Si, du point de vue du propriétaire, de tels actes peuvent à première vue sembler sans danger immédiat pour la sécurité des passagers, ils peuvent cependant avoir des conséquences néfastes à plus ou moins long terme (aussi bien pour le véhicule et ses passagers que pour les finances du constructeur dans le cas de l'installation de pièces de contrefaçon).

Sabotage du véhicule

Cette catégorie regroupe tout ce qui vise à dégrader le fonctionnement du véhicule. Cela consiste à désactiver ou altérer le fonctionnement d'un ou plusieurs ECU, en modifiant leur logiciel embarqué, en émettant des trames malveillantes ou en provoquant un déni de service sur le bus. Les conséquences peuvent aller d'une simple inconvenance (climatisation coupée par exemple) au déclenchement d'un accident potentiellement mortel (freins qui ne répondent plus). Cependant, il est à noter que même un léger dysfonctionnement survenant sur un grand nombre de véhicules pourrait suffire à dégrader durablement l'image de marque d'un constructeur.

Vol de propriété intellectuelle

Un attaquant peut chercher à obtenir des informations confidentielles sur le fonctionnement du véhicule ciblé, par exemple en écoutant et identifiant les trames qui passent sur le bus, voire en récupérant le code source d'un ECU. Au delà d'une utilisation possible de tels procédés par des concurrents (intelligence économique), ces informations peuvent aussi permettre la fabrication et le commerce de contrefaçons d'ECU ou dévoiler des vulnérabilités du matériel à un attaquant.

Vol de données confidentielles

Les voitures embarquant de plus en plus d'équipements informatiques, elles peuvent donc contenir des informations personnelles qu'un attaquant cherchera à récupérer. Par exemple, nous pouvons citer la liste des contacts téléphoniques et l'historique des appels, les coordonnées GPS des derniers trajets ou encore les fréquences radio favorites enregistrées dans l'interface multimédia. D'autres données, circulant sur le réseau embarqué, peuvent aussi rentrer dans cette catégorie, comme la vitesse actuelle du véhicule ou le kilométrage par exemple.

Défi intellectuel

Enfin, de nombreux exemples parsemant l'histoire de l'informatique nous rappellent qu'il ne faut pas exclure l'attaquant motivé simplement par le défi que représente la prise de contrôle d'un véhicule.

2.1.2 Interactions avec un réseau

Comme nous l'avons mentionné précédemment, les réseaux déployés dans les automobiles à l'heure actuelle ont avant tout été pensés en termes de sûreté. En faisant l'hypothèse qu'un attaquant a réussi à s'y connecter, nous nous intéressons maintenant aux possibilités qui lui seraient offertes.

2.1.2.1 Vulnérabilités des bus

Des équipes ont cherché à savoir si des mécanismes de sécurité existaient sur les automobiles actuellement commercialisées. Ces travaux [WWP04], [HKD09] se sont tout d'abord intéressés aux potentielles vulnérabilités existant dans les réseaux internes d'une voiture, en se concentrant majoritairement sur le plus répandu d'entre eux, CAN. Ainsi il est apparu que de par sa conception, le réseau CAN ne permet pas en l'état d'assurer les propriétés de sécurité définies en section 1.3.2.1.

Confidentialité Par construction, tout message envoyé sur le CAN est diffusé (physiquement et logiquement) en clair à tous les nœuds. De fait, un nœud malicieux peut écouter tout le trafic du bus auquel il est connecté.

Disponibilité Les règles d'arbitrage de CAN (voir 1.2.2) font qu'il est aisé pour un attaquant de provoquer un déni de service sur le bus en envoyant en permanence des trames prioritaires, forçant ainsi tous les autres ECU à arrêter leurs émissions.

Intégrité Comme nous l'avons vu précédemment, CAN permet de détecter des altérations accidentelles survenues lors de la transmission d'un message grâce à un CRC. Cependant, cela ne suffit pas en termes de sécurité puisqu'un attaquant peut forger un CRC valide correspondant aux trames qu'il veut émettre. L'intégrité d'un message n'est donc pas assurée.

Authenticité Une trame CAN (voir figure 1.5a) ne contient pas de champ pour l'identification de son expéditeur. Par conséquent, n'importe quel nœud peut en contrôler d'autres sur le même bus en émettant les messages appropriés.

Non répudiation Il n'est pas possible de prouver qu'un nœud a bel et bien émis ou reçu un message, et donc s'il peut être à l'origine d'une attaque sur le réseau.

Ces propriétés de sécurité n'étant pas garanties par les réseaux automobiles actuels (des remarques similaires ont ainsi été émises à propos de FlexRay [NLPJ09]), la prochaine étape est alors de savoir s'il est possible d'exploiter ces lacunes afin de parvenir à prendre le contrôle d'un ECU, voire du réseau dans son ensemble.

2.1.2.2 Actions élémentaires

Comme définie en 1.3.2.2, une attaque est une action malveillante visant à violer une ou plusieurs propriétés de sécurité. Quel que soit l'objectif final de l'attaquant, cette action peut être décomposée en action élémentaires bien définies. Nous considérons que l'attaquant ne peut interagir avec sa cible qu'à travers le ou les réseaux (filaires ou non) auxquels elle est connectée. Si les propriétés de sécurité ne sont pas garanties, les actions élémentaires qu'il peut entreprendre pour arriver à ses fins sont donc comprises dans la liste suivante.

Lecture L'attaquant a accès au trafic réseau et peut voir le contenu de chaque trame.

Interruption L'attaquant empêche un message émis d'atteindre ses destinataires. Ceux-ci ne reçoivent donc aucune information.Il s'agit du but d'une attaque par déni de service.

Modification L'attaquant modifie le contenu d'un message légitimement émis, de sorte que les destinataires ne reçoivent que la version modifiée alors que la source croit avoir émis les bonnes informations.

Fabrication L'attaquant émet lui-même des messages, en les créant de toutes pièces ou bien en rejouant des messages précédemment observés, se faisant alors passer pour un autre nœud du réseau.

Une attaque est dite **active** lorsqu'elle consiste en au moins une action de type interruption, modification ou fabrication, et **passive** si elle ne contient que des actions de type lecture.

2.1.3 Modélisation des attaques

Comme nous l'avons vu en 2.1.1, il existe bien des motivations pour mener des attaques informatiques contre le réseau d'une voiture. Par la suite, nous avons présenté des travaux qui ont montré que de telles attaques étaient réalisables sur des

véhicules actuels. Par conséquent, il est légitime de faire l'hypothèse de l'existence d'attaquants ciblant l'informatique embarquée dans l'automobile. À chacune de ces motivations peut correspondre un ou plusieurs types d'attaquants, possédant différentes ressources (niveau de connaissances, outils, ressources financières), qui auront alors à leur disposition diverses méthodes pour parvenir à leurs fins. Les combinaisons de ces paramètres mènent à une diversité de scénarios d'attaques possibles. En s'inspirant de ce qui se fait dans l'informatique traditionnelle, des travaux comme [BSDT09], [HD07] et [WWW07] ont ainsi cherché à caractériser de telles attaques. Nous reproduisons en figure 2.1 le modèle adapté à l'automobile proposé dans [HD07]. Ce modèle s'inspire, du modèle, défini dans [HL98] et utilisé par les CERT (*Computer Emergency Response Team*) pour leur classification des incidents. Cette classification opère une distinction entre événements, attaques et incidents. Ainsi, un évènement consiste en l'application d'une action contre une cible, ce qui correspond à la partie centrale de la figure. La description d'une attaque prend également en compte les outils utilisés, les vulnérabilités exploitées ainsi que le résultat de ces actions sur le système. Finalement, l'incident correspondant à une attaque est classifié en identifiant l'attaquant ainsi que ses objectifs.

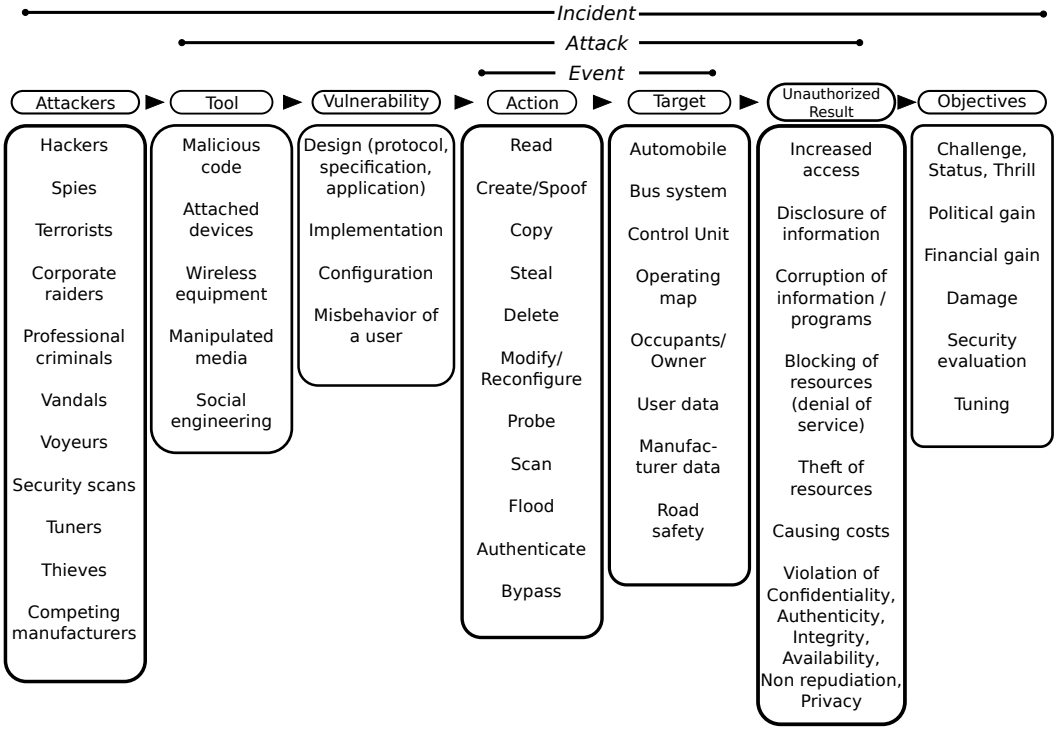


FIGURE 2.1 – Classification du CERT adaptée à un environnement automobile [HD07]

À l'heure actuelle, s'il est possible de trouver de nombreuses illustrations d'attaques contre les réseaux automobiles, comme nous le verrons par la suite, très peu (voire aucun) d'incidents liés à la sécurité concernant des automobiles en circulation

ont été reportés. Pour l'instant, il semble donc difficile de déterminer avec si peu de données si de telles classifications permettent bien de représenter l'ensemble des menaces réelles.

Pour obtenir davantage d'informations au sujet des attaquants, l'utilisation d'un pot de miel adapté à l'automobile a été proposée dans [VNLJ08] (voir la section 3.4.2 pour plus de détails). En faisant circuler des voitures équipées de ces pots de miel pendant une certaine période, cela pourrait permettre de confirmer l'existence d'attaquants ciblant les réseaux automobiles, et le cas échéant donner de plus amples informations sur leurs comportements, leurs moyens, leurs objectifs. Si un déploiement (a fortiori à grande échelle) d'une telle expérience peut sembler irréaliste aujourd'hui, un tel projet pourra devenir réellement intéressant dans un futur à plus ou moins long terme, si les tendances vers l'autonomie et l'interconnexion des véhicules se poursuivent et que de tels véhicules se démocratisent.

2.1.4 Conséquences étendues d'une attaque

Les scénarios d'attaques évoquées précédemment concernent le point de vue de l'attaquant : dans le modèle de la figure 2.1, un incident se conclut ainsi par la réalisation des objectifs de l'attaquant. Or, comparée à un système informatique traditionnel, une voiture moderne constitue un véritable système cyber-physique. Une perturbation (intentionnelle dans le cas d'une attaque) de l'informatique embarquée peut ainsi entraîner des répercussions sur le reste du véhicule et mettre en danger ses passagers. Il convient donc également de considérer le véhicule dans son intégralité lors de l'analyse des effets d'une attaque informatique contre celui-ci, et non juste le système ciblé. [HKD09] définit à cet objet deux catégories de conséquences :

- Les conséquences fonctionnelles, qui décrivent les effets que l'attaque a eu sur les fonctionnalités du système embarqué ciblé. Elles correspondent souvent aux objectifs de l'attaquant, ou sont du moins prises en compte par celui-ci lors de son attaque.
- Les conséquences structurelles, qui décrivent les effets de l'attaque (ainsi que du nouveau comportement du système embarqué ciblé) sur l'ensemble du véhicule ainsi que sur son environnement. Si l'objectif de l'attaque n'était pas *in fine* de nuire aux occupants de la voiture, ces conséquences peuvent ne pas avoir été prévues ou être volontairement ignorées par l'attaquant.

Pour illustrer ces distinctions, considérons deux exemples hypothétiques.

1. Le propriétaire d'un véhicule réussit à manipuler l'odomètre pour diminuer le kilométrage enregistré afin d'en tirer un meilleur prix à la revente. La conséquence fonctionnelle est la diminution de la valeur de la distance parcourue mémorisée. Les conséquences structurelles concerneront tous les effets dus à l'usure réelle du véhicule, usure que le nouveau propriétaire ignore.
2. L'attaquant souhaitant contrôler des ECU sur un bus critique réussit à émettre des trames sur celui-ci, augmentant du coup le trafic sur le réseau. Ces trames

possédant un identifiant bas, les règles d'arbitrage (cf 1.2.2) font que ses messages sont prioritaires sur le bus et retardent donc l'émission d'autres messages, entraînant potentiellement des délais dans l'exécution de fonctions critiques. Ici, les conséquences fonctionnelles seront les réactions provoquées par les trames émises par l'attaquant. Les conséquences structurelles concernent l'impact de la congestion réseau sur le reste des calculateurs du bus (passage en mode dégradé, réactions décalées ...).

2.2 Attaques locales

Dans cette section, nous nous intéressons aux possibilités offertes à un attaquant lorsque celui-ci possède un accès physique direct sur le réseau embarqué, c'est à dire qu'il a le contrôle d'un (ou plusieurs) nœuds du réseau. Nous y décrivons les attaques présentées dans la littérature, réalisables en ayant un tel accès direct sur le réseau embarqué. Nous les avons catégorisées en fonction de l'impact qu'elles ont sur le réseau embarqué du véhicule. Tout d'abord, nous présentons les procédés permettant à un attaquant d'acquérir suffisamment de connaissances sur le réseau avec lequel il cherche à interagir. Ensuite, nous détaillons les attaques n'entraînant pas de conséquences durables sur le réseau embarqué. Finalement, nous nous intéressons aux attaques altérant durablement le réseau, c'est à dire dont l'effet sur le réseau perdure après le passage de l'attaquant.

2.2.1 Découverte de la cible

Tout d'abord, rapppelons que si CAN fait l'objet de standards, ceux-ci régissent le format des trames mais pas l'agencement des données à l'intérieur de chaque champ. Ceux-ci sont ainsi différents pour chaque constructeur, voire pour chaque modèle de véhicule. Ainsi, un même identifiant N ne désignera pas nécessairement le même type de trame d'un modèle à l'autre. Par conséquent, une trame N ne sera pas nécessairement destinée aux mêmes ECUs et les informations contenues dans son champ de données différeront également. Jusqu'à présent, les détails concernant le contenu des trames émises sur le réseau embarqué sont gardés secrets par les constructeurs, qui cherchent ainsi à dissuader toute tentative de manipulation du réseau par autrui.

La première étape d'une attaque va donc généralement consister en une prise de connaissance du système afin de comprendre quel est le rôle de chaque trame. Le réseau CAN (ainsi que les autres protocoles couramment utilisés) ne garantissant pas la confidentialité des données, le simple fait d'être connecté sur un bus suffit pour observer son trafic en clair. Des attaques en "boîte noire" peuvent ainsi être réalisées pour découvrir le fonctionnement du réseau. Cela se fait simplement en se connectant au même bus que le système que l'on souhaite observer, puis en observant le trafic réseau produit en réponse aux stimuli que l'on fournira. Ceux-ci peuvent être des manipulations des différents équipements de la voiture (activer les essuie-glaces, ouvrir une porte ...) ou bien des trames émises sur le bus, qu'elles

soient rejouées suite à une précédente observation ou générées aléatoirement (on parlera alors de *fuzzing*).

Enfin, pour une compréhension plus approfondie du fonctionnement de certains ECU, il reste possible de faire de la rétroingénierie en récupérant le code qu'ils contiennent et en effectuant un débogage sur celui-ci.

Selon les modèles de voiture (et donc en fonction de l'architecture du réseau embarqué), une simple connexion via le port OBD suffit pour pouvoir écouter une part importante du trafic et interagir directement avec de nombreux ECU. [HKD09] a ainsi détaillé un processus d'analyse en boîte noire en réalisant des écoutes via le port OBD, et ce malgré la présence d'un filtre entre le port diagnostic et le reste du réseau.

Par conséquent, si ce type de sécurité par l'obscurité a pu fonctionner un certain temps, l'intérêt grandissant de la communauté informatique au sujet des automobiles modernes condamne son efficacité à plus ou moins long terme par le partage des connaissances acquises à ce sujet. On pensera par exemple aux initiatives telles que OCTANE [EM13], qui fournit une suite logicielle visant à simplifier l'accès aux réseaux embarqués et le partage des données concernant les véhicules analysés.

2.2.2 Prise de contrôle temporaire

Une fois suffisamment de connaissances acquises au sujet du véhicule ciblé, l'émission de trames spécifiques, sélectionnées suite aux observations précédentes, va permettre d'agir directement sur des calculateurs en envoyant des instructions normalement émises par un autre ECU ou dans d'autres conditions. Les exemples dans la littérature commencent à être nombreux et détaillés : si certains anonymisent les résultats de leurs découvertes (en ne dévoilant pas la marque du véhicule et en masquant une partie des trames émises) afin d'éviter une réutilisation potentiellement malveillante de leurs résultats [KCR+10], d'autres [MV13] nomment clairement les modèles des voitures qu'ils ont attaquées et fournissent tous les outils nécessaires pour reproduire et poursuivre leurs travaux. La figure 2.2, tirée de [KCR+10], montre un tableau de bord lors d'une attaque consistant à envoyer des trames contenant des informations erronées le concernant. On y voit ainsi une vitesse erronée (le véhicule étant à l'arrêt) et un message personnalisé est affiché.

Certaines attaques peuvent requérir une mise en place plus sophistiquée, nécessitant l'émission de plusieurs trames différentes à des instants donnés pour fonctionner. Cela est généralement dû à des mécanismes de sûreté de fonctionnement inclus dans les ECU que l'attaquant va chercher à contourner. Ainsi, un exemple tiré de [MV13] consiste à faire croire au véhicule qu'il se trouve dans de bonnes conditions pour activer la fonction de parking automatique pour pouvoir lui faire accepter des trames contrôlant l'inclinaison du volant afin de le forcer à tourner. Concrètement, cela consiste à lui faire croire que le levier est en position de marche arrière et que la vitesse du véhicule est inférieure à 4 miles/h (6,4km/h), soit deux trames (qui plus est sur deux bus distincts) dont l'émission est à coordonner avec celle des messages contrôlant l'inclinaison du volant.

FIGURE 2.2 – Tableau de bord affichant une information vitesse erronée ainsi qu'un message personnalisé [KCR+10]

Ainsi, selon le bus concerné par les attaques, les conséquences peuvent avoir des impacts variables quant à la sécurité des passagers, allant d'une inconvenance mineure (par exemple les vitres électriques qui ne fonctionnent plus ou le volume de la radio bloqué) à des risques mortels (coupure de l'airbag, des freins...) . Cependant, il est à noter que le moindre incident de cet ordre, même en apparence minime, peut causer de forts dommages à l'image de marque d'un constructeur s'il vient à se répéter suffisamment.

2.2.3 Prise de contrôle persistante

Les ECU possèdent un procédé de mise à jour via le réseau. Il est également possible de les reprogrammer [KCR+10], laissant alors le véhicule compromis après le passage de l'attaquant. Cependant, les ECU sont normalement conçus pour accepter de télécharger une nouvelle version de leur logiciel uniquement depuis une source qui s'est correctement authentifiée suite à une séquence défi-response.

Or, ce mécanisme possède actuellement plusieurs défauts. Tout d'abord, les ECU disposant d'une puissance de calcul limitée, la clé utilisée pour le chiffrement est souvent faible (16 bits) et peut donc être découverte par une attaque de type *brute force* pour peu que l'attaquant puisse avoir un accès prolongé à l'ECU concerné. De plus, des erreurs ou négligences dans l'implémentation de ces mécanismes de sécurité peuvent accélérer la découverte de la clé. Par ailleurs, dans les cas étudiés dans [MV13], il apparaît que les clés utilisées pour le chiffrement ne sont pas nécessairement générées aléatoirement mais correspondent au contraire à des noms propres ou autres séquences fréquemment utilisées (du genre *12345*), les rendant donc très vulnérables à une attaque par dictionnaire.

Ainsi, si l'attaquant réussit à faire télécharger son code par un ECU, il obtient alors un accès durable sur le réseau, permettant à son attaque de se poursuivre une fois qu'il n'est plus connecté au véhicule.[NL08] introduit par exemple le concept de virus pour un véhicule, installé dans un ECU lors d'une attaque préalable, mais

qui n'agit que lorsque certaines conditions sont remplies (par exemple en forçant une ouverture des fenêtres lorsque la vitesse lue via le bus dépasse les 100km/h).

La reprogrammation d'un ECU est également utilisée pour propager une attaque sur un autre bus. En effet, comme les architectures réseau actuelles consistent en plusieurs bus reliés entre eux par des ECU passerelles, la compromission d'une passerelle va permettre à l'attaquant d'accéder à une nouvelle portion du réseau. Par conséquent, si un attaquant parvient à obtenir un accès sur le réseau, les architectures actuelles font qu'il peut à terme obtenir un contrôle total sur les ECU du véhicule.

Cependant, il peut être objecté que les attaques mentionnées précédemment impliquent qu'un attaquant ait déjà obtenu un accès physique au réseau interne du véhicule. Il existait donc déjà une faille de sécurité en amont puisque l'attaquant a pu pénétrer dans la voiture pour se connecter sur son réseau. De plus, il est dans ce cas également possible de nuire en effectuant plus simplement et plus rapidement des attaques mécaniques sur le véhicule (par exemple en déconnectant les freins au lieu d'attaquer électroniquement les calculateurs correspondants).

2.3 Attaques à distance

Les nombreux moyens de communication inclus dans les véhicules modernes (voir figure 2.3) font que les réseaux embarqués de tels véhicules ne peuvent plus être considérés comme fermés (ou isolés) car ils fournissent à l'attaquant une porte d'entrée potentielle pour mener une attaque à distance.

Ainsi, en 2011, Chechoway et al. [CMK+11] ont montré qu'il était possible de reproduire les attaques décrites dans [KCR+10] sur un véhicule récent en attaquant au préalable les nombreuses interfaces du véhicule permettant de communiquer avec une source de données externe, prouvant ainsi qu'un accès physique n'était plus requis pour prendre le contrôle du réseau interne de la voiture. Ces attaques peuvent être classées selon leur portée : accès physique indirect, accès sans fil de courte portée et accès sans fil de longue portée. Nous les décrivons par la suite.

Par ailleurs, comme mentionné au début de la section 2.1, il existe d'autres type d'attaques à distance contre une voiture : ce sont les cas où l'attaquant ne cherche pas spécifiquement à obtenir un accès sur le bus. En d'autres termes, la cible finale de son attaque est le système avec lequel il communique à distance, que ce soit par exemple pour écouter les conversations via le système multimédia ou pour perturber des capteurs utilisant des technologies de communication sans fil. Des exemples de telles attaques tirés de la littérature sont également mentionnées dans cette partie, également classées en fonction de leur portée.

2.3.1 Accès physique indirect

Par accès physique indirect, nous désignons les cas où l'attaque porte dans un premier temps sur un élément extérieur à la voiture. C'est la connexion de cet élément avec le réseau qui permet alors à l'attaque d'atteindre le réseau interne. Ainsi,

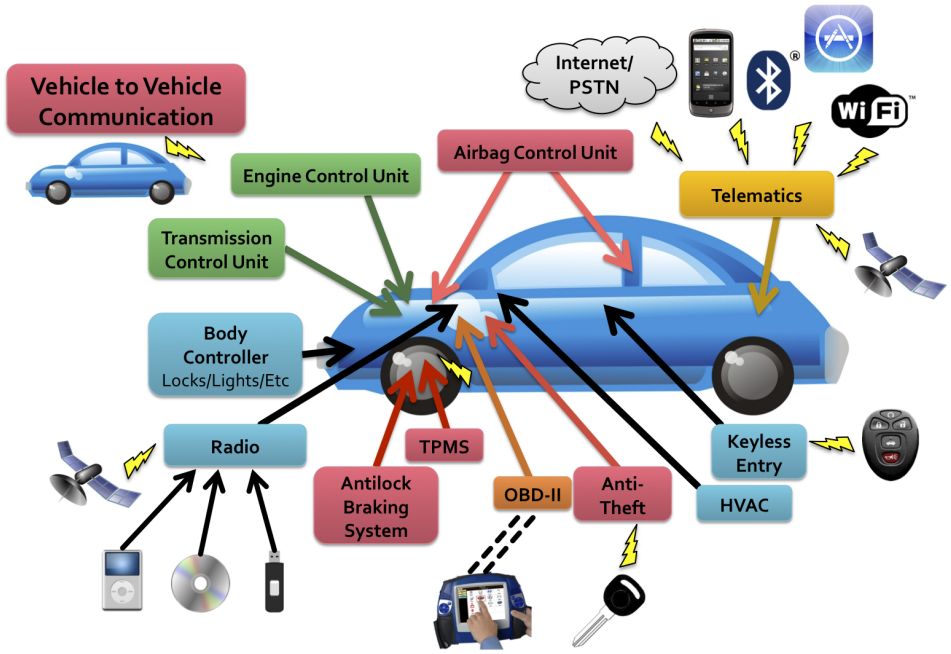


FIGURE 2.3 – Canaux de communications typiques d'une automobile moderne et potentielles cibles critiques du réseau interne (tirée de [CMK+11])

si un accès physique au réseau interne est toujours requis pour finaliser l'attaque (il reste nécessaire de pénétrer dans le véhicule), cette étape n'est plus réalisée par l'attaquant mais par un tiers à son insu. En se plaçant du point de vue de l'attaquant, nous considérons de telles attaques comme des attaques à distance.

Outils de diagnostic

La section 2.2 a montré qu'il était possible d'utiliser le port de diagnostic pour mener des attaques contre le réseau. Ici, nous envisageons en fait les attaques dirigées contre des outils destinés à être connectés au port de diagnostic. Dans [CMK+11], un tel outil, connecté sur le port et piloté en Wi-Fi depuis un ordinateur a ainsi pu être attaqué. Depuis un ordinateur connecté sur le même réseau que l'outil, une injection de code à pu être réalisée suite à des failles dans l'implémentation des protocoles de communication utilisés par ce dernier. Dès lors, le module de diagnostic injecte des paquets malveillants dans le bus CAN quand il se retrouve connecté à un port OBD. Pire, il pouvait à son tour infecter d'autres modules partageant le réseau Wi-Fi. Si cet exemple concerne un outil prévu pour le diagnostic, il peut aisément être étendu à tout appareil prévu pour être branché sur le port OBD. Il existe ainsi des dongles OBD communiquant en bluetooth ou Wi-Fi avec un smartphone, permettant à l'utilisateur de récupérer et visualiser des données circulant sur le CAN [1]. De même, certaines compagnies d'assurance offrent

1. `https://www.automatic.com`

des réductions à leurs adhérents si ceux-ci acceptent d'enregistrer leur trafic CAN à l'aide d'un dispositif équivalent pour prouver qu'ils conduisent prudemment [2]. Il suffit alors qu'une vulnérabilité soit découverte [3] sur de tels dispositifs pour ouvrir la porte à des attaques de ce genre.

Lecteur CD

Dans [CMK+11], deux exemples d'attaques mettant en jeu le lecteur CD d'un véhicule ont été présentés. Tout d'abord, l'équipe a identifié une fonctionnalité permettant de mettre à jour cet ECU par CD-ROM. Si le CD contient un fichier nommé d'une certaine manière (non décrite), un message est alors brièvement affiché sur l'écran et si le conducteur n'appuie pas rapidement sur le bon bouton, une reprogrammation de l'ECU se déclenche. Il est par ailleurs à noter que cette technique de mise à jour n'est pas celle employée par le constructeur et que sa présence témoigne probablement d'une réutilisation de code d'un projet précédent par le fournisseur. Une autre vulnérabilité a été identifiée dans le décodage des fichiers WMA par le lecteur. Une analyse approfondie a permis la création d'un fichier audio en apparence normal (et parfaitement lisible sur un ordinateur) mais qui, en exploitant un dépassement de tampon, provoque l'émission de trames CAN par le système multimédia de la voiture lorsque celui-ci le joue.

Si la probabilité d'une attaque correspondant au premier exemple reste faible (il faut que le conducteur introduise un CD inconnu dans son lecteur sans avoir au préalable cherché à identifier son contenu), celui-ci illustre en revanche les risques liés à la réutilisation de composants (matériels ou logiciels) sans vérification poussée. En revanche, il est tout à fait possible d'imaginer un scénario d'attaque correspondant au deuxième exemple, où un attaquant va chercher à compromettre indistinctement le plus grand nombre possible de véhicules utilisant ce système multimédia en mettant à disposition des fichiers malveillants sur les réseaux de partage en ligne.

Prise USB

Plusieurs situations sont envisageables. Tout d'abord, un cas similaire à celui du lecteur CD est possible avec un lecteur multimédia embarqué qui accède à un fichier corrompu présent sur une clé usb. Une autre possibilité concernerait un smartphone infecté connecté en USB. Si une attaque de ce type n'a pas encore été documentée, les précédents concernant la sécurité des interfaces de connexion d'une voiture couplés aux possibles attaques contre un téléphone (par exemple via bluetooth [4] ou suite au téléchargement d'une application malveillante) rendent cette hypothèse plausible.

2. http://www.progressive.com/auto/snapshot/
3. http://www.forbes.com/sites/thomasbrewster/2015/01/15/researcher-says-progressive-insurance-dongle-totally-insecure/
4. http://trifinite.org/trifinite_stuff_bluebug.html

2.3.2 Attaques de courte portée

Cette section évoque de potentielles attaques utilisant les moyens de communication sans fil à courte portée (c'est-à-dire de l'ordre du mètre) que peut posséder un véhicule.

Connexion sans fil d'équipements mobiles

Les véhicules modernes peuvent permettre un appariement avec des équipements mobiles. Par exemple, le conducteur peut connecter son téléphone via Bluetooth ou Wi-Fi et utiliser les enceintes de sa voiture comme un kit main libres. Or, il peut arriver que l'implémentation de ces protocoles comporte des failles. Celles-ci peuvent se prêter à des scénarios d'attaques directes où un attaquant muni d'un terminal proche de la voiture tente de prendre le contrôle du module gérant ces connexions, ou indirectes si elles ciblent dans un premier lieu un équipement tiers préalablement autorisé à dialoguer avec ces équipements (tel que le smartphone du conducteur).

[CMK+11] décrit ainsi plusieurs attaques exploitant une vulnérabilité (là aussi, un dépassement de tampon) présente dans la gestion du Bluetooth par l'unité télématique de leur véhicule cible. Cependant, l'exploitation de cette faille requiert d'être capable d'appairer un appareil avec le véhicule cible.

Le premier exemple consiste ainsi en une attaque indirecte, où une application mobile de type cheval de Troie surveille les connexions Bluetooth du téléphone. Comme précédemment, le scénario d'attaque implique que le propriétaire du véhicule cible a téléchargé à son insu le programme sur son smartphone. Lorsque celui-ci se retrouve appairé avec le système télématique du véhicule, l'application déclenche alors son attaque et prend le contrôle de l'ECU.

Le second exemple est en revanche une attaque directe dont la première étape consiste à identifier l'adresse MAC (Bluetooth) de la voiture puis à s'appairer avec elle. Or, il s'est avéré que la voiture pouvait accepter les requêtes d'appairage sans requérir d'intervention du conducteur, ce qui a rendu possible une attaque par force brute afin de trouver le mot de passe généré par la voiture pour confirmer l'appairage. Il est à noter que le temps requis pour une attaque de la sorte peut être supérieur à 10h, ce qui implique que l'attaquant puisse rester à proximité du véhicule ciblé pendant une longue durée. Cependant, le processus peut être multiplexé, ce qui peut permettre d'attaquer plusieurs voitures proches en simultané (par exemple sur un parking) pour augmenter les chances de réussite dans le cas où l'attaquant ne cherche pas à cibler un véhicule précis.

L'exploitation de telles attaques peut permettre la récupération de données stockées dans le module de communication, l'écoute des communications passées ou là encore la prise de contrôle du réseau interne si le module possède un accès à celui-ci (c'est généralement le cas).

Communications Car2Car

Les communications entre un véhicule et d'autres véhicules ou des équipements de bord de route constituent une des prochaines grandes évolutions dans le domaine des transports. En effet, une automobile sera ainsi capable de communiquer son état aux voitures environnantes. Cela permettra par exemple au véhicule d'alerter le conducteur en cas de danger imminent (changement dans la limitation de vitesse, véhicules arrivant dans un carrefour sans visibilité, freinage d'urgence...), voire de réagir automatiquement. Parmi les risques identifiés [MLLS12], nous pouvons par exemple citer l'espionnage des communications entre véhicules, l'envoi de fausses informations à un véhicule afin de déclencher une réaction inappropriée de sa part, que ce soit afin de nuire à la cible (en ne la faisant pas freiner par exemple) ou pour en retirer un bénéfice (en se faisant passer pour un véhicule prioritaire). Enfin, nous envisageons également les cas concernant la prise de contrôle de l'ECU responsable des communications car2car suite à l'exploitation d'une faille de sécurité dans celui-ci.

TPMS

TPMS, pour *Tire Pressure Monitoring System* (système de surveillance de la pression des pneus) est constitué d'un capteur de pression situé à l'intérieur du pneu qui envoie ses mesures à un ECU dédié du réseau embarqué via un émetteur radio courte portée. Les TPMS sont désormais obligatoires aux États-Unis, en Europe et au Japon. Dans [RMM+10], des attaques ciblant le TPMS ont montré que l'on pouvait écouter les informations émises par celui-ci depuis des distances allant jusqu'à 40 mètres, ce qui permettait de tracer le véhicule, mais qu'il était également possible d'envoyer des messages malveillants contenant des informations erronées à l'ECU récepteur. En conséquence, le voyant signalant une pression des pneus anormale s'allumait alors qu'aucun danger réel n'était présent.

Ouverture à distance

De nombreux véhicules sont maintenant équipés d'un déverrouillage à distance de leurs portes. Bien que les signaux correspondants soient chiffrés, il est possible de trouver les clés ou contourner ce mécanisme de chiffrement. Par exemple, nous pouvons citer des attaques permettant de récupérer la clé utilisée par le système de chiffrement Keeloq, employé par plusieurs constructeurs [CBW08], [EKM+08]. Par ailleurs, Francillon et al. [FDC10] ont décrit comment ouvrir et démarrer à l'insu de leur propriétaire des voitures (dix modèles différents) équipées d'un système PKES en effectuant une attaque par relais. En plaçant une antenne proche (environ 8m) du propriétaire du véhicule ciblé, ils réussirent à transmettre le signal émis par la clé sur une distance de 50m, faisant ainsi croire à la voiture que ses clés se situaient à proximité. Ce dernier exemple montre également que la notion de courte portée est relative puisque la portée théoriquement courte de certaines connexions peut être amplifiée en utilisant des antennes plus puissantes ou en relayant le signal. Ainsi des

exemples d'attaques via bluetooth [5] ont été réalisés depuis des distances dépassant le kilomètre grâce à l'utilisation d'antennes possédant un gain élevé.

2.3.3 Attaques de longue portée

La dernière catégorie concerne les attaques pouvant être réalisées via des technologies de communication à longue portée, typiquement via les réseaux cellulaires, et permettant donc à un attaquant de se situer virtuellement à n'importe quelle distance du véhicule cible.

Téléphonie

Suite à l'identification de vulnérabilités dans le module télématique, [CMK+11] décrit une attaque où suite à un appel passé à la voiture (équipée d'une carte SIM 3G), l'exploitation d'une série de vulnérabilités a permis de forcer l'unité télématique à télécharger et exécuter du code via le réseau 3G, compromettant effectivement l'automobile.

Navigation web

Dans le cas où un véhicule intègrerait un navigateur web, des exploitations des vulnérabilités potentielles de celui-ci sont parfaitement envisageables, d'une manière similaire à ce qui peut toucher les ordinateurs traditionnels et les terminaux mobiles. Là aussi, les conséquences peuvent être similaires aux attaques évoquées précédemment : accès aux données stockées dans le module accueillant le navigateur ainsi qu'un accès potentiel aux bus auquel il est connecté.

2.3.4 Attaques indirectes à longue portée

Ici, nous décrivons des cas où les attaques ont lieu sans que l'attaquant ait eu à s'approcher de sa cible ni qu'aucune connexion physique ne soit requise, mais pour lesquels un ou des intermédiaires doivent cependant être préalablement attaqués.

Boutique en ligne

Dans une tendance équivalente à ce qui s'est produit sur les smartphones, plusieurs constructeurs automobiles souhaitent mettre à disposition de leurs usagers via une unique boutique en ligne (similaire à l'Appstore d'Apple ou au Google Play Store) un choix d'applications installables dans leur véhicule. Une attaque réussie contre la boutique en ligne, ou plus probablement la mise à disposition d'une application malveillante (de type cheval de Troie) sur une telle boutique, que ce soit à cause d'une faille de sécurité dans une application légitime ou car un développeur a réussi à contourner les vérifications nécessaires à la mise sur le marché (de tels

5. `http://trifinite.org/trifinite_stuff_lds.html`

exemples se sont produits sur l'Appstore comme sur le playstore [6]) pourraient avoir un impact sévère et à grande échelle.

Déclenchement par canaux auxiliaires

Dans [CMK+11], un scénario hypothétique est évoqué où après avoir pris le contrôle d'un véhicule par un des biais décrits précédemment, une sorte de porte dérobée est mise en place dans un calculateur. Dès lors, lorsque le véhicule reçoit un certain signal, une série d'instructions est exécutée par les calculateurs compromis. Une telle technique couplée à un grand nombre de véhicules préalablement infectés ouvre la porte à des attaques de grande envergure aux conséquences potentiellement désastreuses. Il est intéressant de noter que la porte d'entrée initiale de l'attaquant n'influe pas nécessairement sur la portée du signal déclencheur. Ainsi, même si un autoradio/système multimédia dépourvu de communications 3G est compromis via une attaque locale indirecte, le déclenchement de la deuxième partie de l'attaque pourrait par exemple se faire via RDS (*Radio Data System*, un protocole utilisé pour inclure des données dans les signaux FM).

2.4 Classification des attaques

Les différentes attaques présentés dans ce chapitre nous ont mené à en proposer une classification. Celle-ci cherche à mettre en évidence la diversité des scénarios possibles et des menaces devant être prises en compte lors de la conception des architectures réseaux embarquées dans les automobiles actuelles. Cette classification est illustrée en figure 2.4. Nous avons ainsi identifié quatre axes permettant de catégoriser les attaques ciblant les systèmes embarqués d'une automobile :

Niveau d'interaction avec le système Nous distinguons trois niveaux d'interaction possibles (n'étant pas mutuellement exclusifs). Tout d'abord, l'attaquant peut interagir avec le matériel, que ce soit dans le cadre d'attaques physiques contre le matériel lui-même, ou encore en brouillant les signaux reçus par des capteurs. Une attaque peut également agir sur le logiciel embarqué dans un ECU. Enfin, une attaque peut affecter le réseau interne du véhicule.

Portée Comme nous l'avons vu en 2.3, les différents moyens de communications inclus dans une automobile moderne font qu'il peut être vulnérable à des attaques menées via une connexion physique directe, ou à distance via des protocoles de communication à courte ou longue portée.

Lien Un attaquant peut interagir directement ou indirectement avec le véhicule qu'il cible. Dans le premier cas, son attaque est dirigée uniquement contre les systèmes embarqués d'un véhicule. Dans le second cas, l'attaquant cible préalablement

6. voir par exemple `http://www.securelist.com/en/blog/208193641/Find_and_Call_Leak_and_Spam`

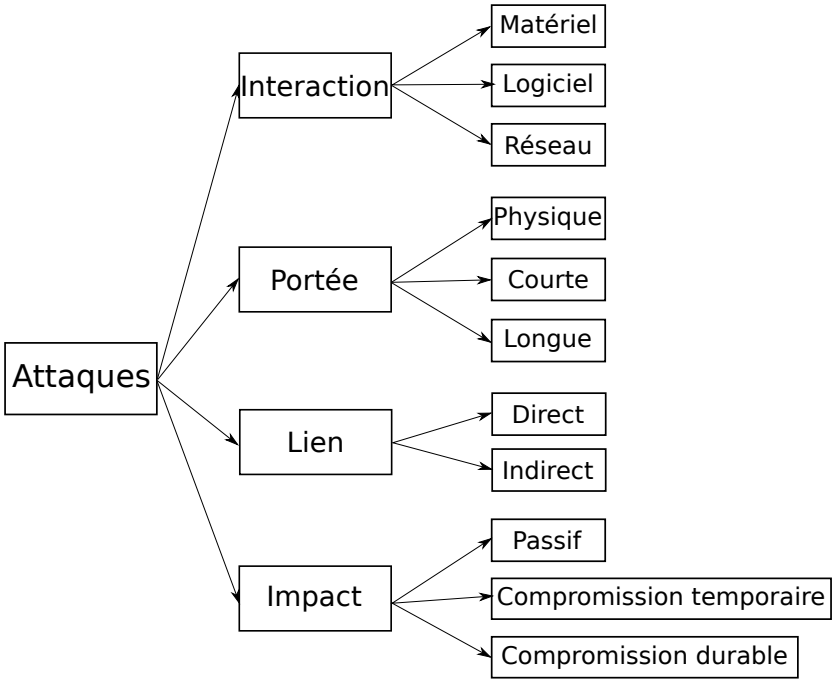


FIGURE 2.4 – Classification des attaques ciblant les calculateurs embarqués dans une automobile

un dispositif qui communiquera avec le véhicule et servira donc de relais pour finaliser l'attaque.

Impact de l'attaque Nous caractérisons l'impact d'une attaque sur un système embarqué sur trois niveaux. L'impact le plus faible est qualifié de passif, c'est à dire que l'attaque n'a pas altéré le fonctionnement du système ciblé. Par exemple, une attaque visant à s'identifier au près du dispositif Bluetooth d'une voiture afin d'accéder aux données qu'il contient rentre dans cette catégorie. De telles attaques peuvent être difficiles à détecter mais n'entraînent pas de risques majeurs pour l'intégrité du véhicule. Une compromission temporaire désigne un cas où l'attaquant va activement perturber le fonctionnement d'un système embarqué, mais dont les effets néfastes cessent une fois l'attaque finie : si un attaquant souhaite effectuer une nouvelle attaque, il devra reproduire la même démarche intégralement. Par exemple, les attaques présentées précédemment consistant en un rejeu de trafic réseau envoyé par le port OBD rentrent dans cette catégorie. Enfin, une compromission durable désigne les cas où une attaque compromet durablement un système embarqué. Les effets de l'attaque persistent même lorsque l'attaquant n'est plus connecté au véhicule. Une reprogrammation non-autorisée d'un ECU ou l'installation d'une porte dérobée dans un système multimédia constituent ainsi des exemples de compromissions durables.

2.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux attaques pouvant cibler les calculateurs embarqués des automobiles modernes. Nous nous sommes tout d'abord attachés à analyser la structure de telles attaques. Nous avons ainsi énuméré les possibles objectifs pouvant motiver une attaque contre le réseau embarqué, Puis, après avoir présenté les lacunes en terme de sécurité des protocoles de communication actuellement utilisés, nous avons décrit les actions élémentaires effectuées lors d'une attaque sur le réseau embarqué. A partir de ces informations, il est alors possible de modéliser un scénario d'attaque, comme illustré par un exemple inspiré par le modèle utilisé par les CERT. Finalement, nous avons montré qu'une attaque menée contre un système cyber-physique comme une voiture peut avoir des conséquences qui s'étendent au delà du scénario initialement prévu par l'attaquant, affectant potentiellement la sûreté du véhicule à plus ou moins long terme. Si un attaquant peut choisir de les ignorer, elles ne sont pas à négliger lors de l'évaluation des risques liés à une attaque.

Par suite, nous avons présenté les différentes attaques possibles, illustrées par des exemples de la littérature. Nous nous sommes d'abord intéressés à ce qu'il était possible de faire avec un accès au réseau embarqué, en opérant une distinction entre les attaques visant à acquérir des informations sur le réseau, les attaques visant à une prise de contrôle temporaire du véhicule, puis celles débouchant sur une compromission durable du réseau. Enfin, nous avons décrit différentes attaques pouvant permettre d'obtenir un accès au réseau interne d'une automobile à distance, classées en fonction de leur portée.

Finalement, nous avons présenté une classification des scénarios d'attaque contre les systèmes embarqués d'une automobile selon quatre axes. Les attaques ciblant les calculateurs embarqués peuvent ainsi être catégorisées en fonction de leur niveau d'interaction avec le réseau, de leur portée, du lien entre l'attaquant et le réseau (attaque directe ou indirecte), et enfin de leur impact sur le réseau interne. La conception de politiques de sécurité pour les réseaux automobiles doit ainsi tenir compte de cette diversité de menaces afin de proposer des mécanismes de défense adaptés.

Chapitre 3

Mécanismes de sécurité pour les architectures automobiles embarquées : état de l'art

Sommaire

Certaines des attaques décrites dans le chapitre précédent sont possibles à cause de mauvaises implémentations des protocoles ciblés, de vulnérabilités présentes dans le code des applications concernées (permettant des débordement de tampon) voire à cause du non-respect des spécifications du système. Par conséquent, de telles attaques peuvent être théoriquement évitées en respectant strictement de bonnes pratiques de programmation ainsi qu'en suivant les recommandations de sécurité existantes quant à l'implémentation de certains protocoles (voir par exemple [SP08]). Cependant, la complexité croissante des systèmes embarqués couplée aux contraintes de la conception automobile (que nous avons évoquées en 1.1.1) rendent potentiellement impossible la vérification stricte de l'intégralité du code présent dans un véhicule. Par conséquent, l'intégration de mécanismes de sécurité supplémentaires est nécessaire pour garantir la sécurité des communications au sein d'une automobile.

Dans ce chapitre, nous présentons un tour d'horizon des implémentations de mécanismes de sécurité dans l'automobile. Nous évoquerons tout d'abord les stratégies globales guidant le développement et l'implémentation de mécanismes de sécurité dans les réseaux automobiles. Nous présentons tout d'abord les projets européens,

ayant trait à la sécurité informatique pour les automobiles. Faisant coopérer industriels et académiques sur des sujets porteurs, notamment les communications V2X, ils visent à proposer des standards de conception tenant compte des problématiques de sécurité-immunité pour les véhicules futurs. Nous présenterons ensuite les différentes techniques de sécurité informatique appliquées (ou applicables) à l'automobile, illustrées par des exemples de la littérature. Nous nous intéresserons tout d'abord aux mécanismes dits préventifs, puis décrirons enfin les mécanismes réactifs.

3.1 Projets européens

Pour les constructeurs, la prise de conscience de la nécessité de développer, d'inclure et d'harmoniser les solutions de sécurité dans l'automobile est désormais bien réelle. Ainsi, de nombreux projets à grande échelle associant constructeurs, fournisseurs et académiques ont vu le jour afin de répondre aux problématiques de sécurité soulevées par les automobiles modernes. Nous présentons ci-après des exemples de projets européens (classés par ordre chronologique) qui, en se concentrant sur différents aspects de la sécurité des communications automobiles, contribuent à la mise en place de mécanismes de défense adaptés aux véhicules.

Sevecom

Le projet Sevecom [1] (***Se****cure* ***Ve****hicular* ***Com****munication*), de janvier 2006 à décembre 2008, se concentrait sur les communications V2X. Il a permis d'établir des spécifications concernant la sécurité de telles communications pour éviter l'envoi, l'écoute ou la modification de messages par un attaquant.

PRECIOSA

PRECIOSA [2] (***PR****ivacy* ***E****nabled* ***C****apability* ***I****n Co-****O****perative Systems and* ***S****afety* ***A****pplications*), de mars 2008 à mars 2010, s'est concentré sur les problématiques liées au respect de la vie privée des conducteurs dans le cadre des communications V2X. En effet, ces communications nécessitent à la fois un suivi spatial et temporel des véhicules concernés. Les objectifs du projet consistaient donc à définir des méthodes permettant de garantir la confidentialité de ces données privées lors de leur utilisation ou de leur stockage.

EVITA

EVITA [3] (***E****-Safety* ***V****ehicle* ***I****ntrusion Pro****t****ected* ***A****pplications*), de juillet 2008 à décembre 2011, portait sur la conception d'une base (matérielle et logicielle) de

1. `http://www.sevecom.org`, désormais inaccessible, mais visionnable sur `http://web.archive.org/web/20130603042821/http://www.sevecom.org/`
2. `http://www.preciosa-project.org`
3. `http://www.evita-project.org`

confiance pour les architectures de sécurité des réseaux automobiles embarqués. Le projet a débouché sur des spécifications de modules de sécurité, incluant des solutions logicielles et matérielles afin de sécuriser les informations échangées entre ECUs, ainsi que de protéger ceux-ci contre le vol ou des manipulations non autorisées. Nous revenons sur les résultats de ce projet par la suite, en 3.3.1.3

OVERSEE

L'objectif d'OVERSEE [4] (***O**pen **Ve**hicular **Se**cur**e** platform*), de janvier 2010 à juin 2012, avait pour objectif de concevoir une plate-forme logicielle standardisée et ouverte permettant l'exécution sécurisée de plusieurs applications d'origine et de criticités différentes au sein d'un même système multimédia. La solution retenue est présentée plus en détail en 3.3.2.2.

PRESERVE

L'objectif de PRESERVE [5] (*Preparing Secure Vehicle-to-X Communication Systems*) est de poursuivre les avancées des projets précédents et de les combiner afin de proposer une implémentation concrète d'une architecture (protocoles et système embarqué) de sécurité pour les communications V2X, adaptée aux contraintes du marché et utilisable directement pour de futures expérimentations. Ce projet a commencé en janvier 2011 et sa fin est prévue pour juin 2015.

3.2 Défense en profondeur

Une automobile est un système complexe, et un réseau d'automobiles communiquant entre elles l'est d'autant plus. Pour être véritablement efficace, la mise en place de mécanismes de sécurité dans automobile doit se faire avec une vision d'ensemble du (ou des) système(s) concerné(s). À l'origine, la défense en profondeur (*defense-in-depth*) est une stratégie militaire dont le but est de ralentir le plus possible l'avancée d'un ennemi plutôt que de chercher à stopper immédiatement l'assaut en renforçant la première ligne.

En informatique, la défense en profondeur [NSA12] consiste à multiplier les mécanismes de sécurité pour protéger l'ensemble d'un système. En pratique, cela revient à déployer des mécanismes de sécurité (complémentaires et/ou redondants) à plusieurs niveaux du système, au lieu de simplement protéger ses points d'entrée. L'objectif est ainsi de réduire les risques lorsqu'un de ces mécanismes devient compromis ou du moins n'a pas suffi à arrêter une attaque. Ainsi, une politique de sécurité incluant des concepts de défense en profondeur permet de faire face à une plus grande diversité de menaces et d'augmenter la tolérance aux intrusions d'un système. Cependant, il est évidemment nécessaire d'obtenir un bon équilibre entre les capacités de protection déployées, leur coût et les performances du système, sans

4. `http://www.oversee-project.com`
5. `http://www.preserve-project.eu`

quoi l'accumulation de mesures de protection finirait par s'avérer contre-productive. Par exemple, obliger un utilisateur à utiliser plusieurs mots de passe à la suite pourrait mener celui-ci à les noter par écrit pour pouvoir les retenir, ce qui augmente les risques de sécurité au lieu de les réduire.

En 1995, Halme et al. [HB95] ont défini six catégories de techniques pour se défendre contre les intrusions :

1. Prévention : l'ensemble des moyens déployés pour contrer des tentatives d'intrusion spécifiques, allant de l'intégration de concepts de sécurité dès l'étape de spécification du système à l'utilisation d'outils d'analyse de vulnérabilités, d'antivirus ou encore de pare-feux.
2. Préemption : les mesures prises en amont de toute attaque possible. Elles incluent aussi bien l'éducation des utilisateurs ou des concepteurs du système à protéger que l'obtention d'informations au sein des communautés d'attaquants potentiels.
3. Dissuasion : l'ensemble des techniques visant à diminuer (réellement ou en apparence) l'intérêt d'une attaque en jouant sur le ratio risque/récompense.
4. Diversion : le fait de faire croire à un attaquant qu'il a réussi à s'introduire dans le système alors qu'il est en réalité en train d'interagir avec un leurre.
5. Détection : les techniques cherchant à distinguer une utilisation frauduleuse au milieu d'usages légitimes du système.
6. Contre-mesure : la capacité du système à réagir automatiquement en cas d'intrusion avérée.

En fonction du système à protéger et des politiques de sécurité mises en place, une défense en profondeur adaptée choisira plusieurs moyens appartenant à des catégories différentes (mais pas forcément à toutes) pour minimiser la probabilité et les conséquences d'une intrusion réussie. La figure 3.1 illustre ainsi ce principe avec l'intégralité de ces catégories. Sur cette figure, la flèche horizontale correspond aux progressions des attaques au cœur du système et les rectangles correspondent aux proportions de ces attaques que chacun des moyens doit traiter. Notons qu'un moyen peut être appliqué lors de plusieurs étapes (il peut par exemple y avoir des mécanismes de prévention situés à l'extérieur et à l'intérieur du système). De plus, si un de ces moyens s'avère très efficace, alors les moyens qui suivent ont alors moins d'attaques à traiter. Cela se traduit sur la figure par des rectangles de taille décroissante. Par ailleurs, en complément de ces moyens, des analyses légales, ou *forensics*, peuvent être effectuées après une attaque réussie (ou du moins dont on suspecte la réussite) pour permettre de retracer les évènements s'étant produits afin d'attester ou non qu'une attaque a eu lieu et identifier ses causes, ses conséquences et son déroulement.

En ce qui concerne les risques d'attaques informatiques dans l'automobile, une approche de type défense en profondeur apparaît adaptée. En effet, comme vu en 1.1.3.2, une automobile moderne communique au sein de plusieurs réseaux interconnectés qui peuvent posséder des niveaux de confiance variés. Ainsi pour synthétiser leurs travaux dans le domaine de la sécurité des réseaux automobiles, Nilsson

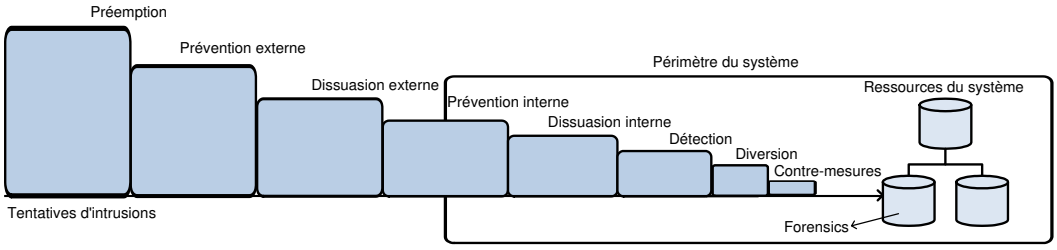


FIGURE 3.1 – Défense en profondeur, d'après [HB95]

et al. ont proposé un modèle de défense en profondeur dans [NL09] appliqué aux communications V2X. Dans ce cas, leur modèle ne reprend pas l'intégralité des moyens décrits dans [HB95] mais n'inclut que des moyens de prévention, détection et diversion ainsi que des analyses *forensics*.

La suite de ce chapitre est dédiée à la présentation des techniques de sécurité, existantes ou en cours de développement, adaptées aux réseaux automobiles ou pour lesquelles une adaptation a été envisagée. Dans le chapitre 2, nous nous sommes focalisés sur les attaques informatiques. De même, ici, nous nous concentrons sur les mécanismes de défense concernant les attaques ciblant les logiciels et/ou les réseaux embarqués. Nous les groupons en deux grands ensembles :

- Les mécanismes de sécurité préventifs, qui se visent à empêcher une attaque de pénétrer dans le système ou dans un sous-système
- Les mécanismes réactifs, qui vont analyser le comportement d'un attaquant afin de pouvoir riposter, immédiatement ou lors d'une intrusion future.

3.3 Techniques de sécurité préventives

Comme expliqué en 3.2, l'étape de prévention vise à empêcher un attaquant de pénétrer à l'intérieur d'un système en renforçant la sécurité du système en prévision d'une attaque. Les techniques présentées ici sont donc utilisées pour corriger les failles de sécurité qu'un attaquant pourrait exploiter pour parvenir à ses fins.

Nous avons regroupé celles-ci selon 3 catégories :

- Le chiffrement des communications
- La sécurisation des données logicielles
- L'architecture du réseau

Pour chacune de ces catégories, nous précisons contre quels types d'attaques de notre classification présentée en 2.4 ces mécanismes sont adaptés.

3.3.1 Sécurité des communications

Nous nous intéressons ici aux techniques visant à sécuriser les canaux de communications utilisés par une automobile, qu'ils soient internes (communications entre ECUs) ou externes (communications V2X, mais aussi avec un appareil électronique de l'utilisateur par exemple). Ces techniques ont recours à la cryptographie pour

chiffrer ou signer les informations échangées via ces canaux, afin de garantir l'authenticité, l'intégrité et la confidentialité de celles-ci. Les moyens de sécurisation des communications visent à se prémunir contre des attaques intégrant une interaction directe avec le réseau. Si l'attaquant ne peut pas déjouer les mécanismes de chiffrement, il ne pourra ni lire, ni modifier les données du système. En revanche, dans le cadre d'attaques indirectes, de tels mécanismes peuvent s'avérer insuffisant si le vecteur utilisé pour finaliser l'attaque possède déjà les capacités de communication nécessaires avec le véhicule.

3.3.1.1 Principes généraux de cryptographie

Il existe deux grandes catégories de chiffrement : la cryptographie symétrique (ou à clé secrète) et la cryptographie asymétrique (ou à clé publique).

- Dans les systèmes symétriques, l'émetteur et le récepteur possèdent tous deux la même clé (secrète), qui servira aussi bien à chiffrer qu'à déchiffrer les messages qu'ils s'échangent. Tant que la clé reste secrète, la confidentialité des messages est alors garantie. AES(*Advanced Encryption Standard*) est un exemple d'algorithme de chiffrement à clé secrète utilisé aujourd'hui.
- La cryptographie asymétrique, elle, se base sur un couple clé publique/clé privée. Un message codé avec l'une ne pourra être décodé qu'en utilisant l'autre. La clé publique peut ainsi être diffusée alors que la clé privée doit être gardée secrète. Un expéditeur souhaitant envoyer un message confidentiel à un destinataire particulier code son message avec la clé publique du destinataire. Ce dernier, seul possesseur de la clé privée correspondante, est alors capable de le déchiffrer. Inversement, un expéditeur peut signer un message avec sa clé privée, ce qui lui permet de s'authentifier auprès des destinataires qui vérifieront le message avec sa clé publique. RSA est un algorithme de chiffrement asymétrique.

Si la cryptographie asymétrique ne nécessite pas de secret partagé entre les participants, un algorithme de chiffrement à clé publique nécessitera plus de puissance de calcul (algorithmes plus complexes) et plus de mémoire (clé plus longue) qu'un chiffrement à clé secrète pour un niveau de sécurité équivalent contre des attaques visant à découvrir la clé secrète.

3.3.1.2 Application dans l'automobile

Tout d'abord, il est à noter que des mécanismes de sécurité faisant appel à la cryptographie existent déjà dans les véhicules actuels. C'est par exemple le cas de systèmes d'ouverture à distance, basés sur un système de type défi-réponse qui doit permettre à la voiture de s'assurer que le signal a été émis par la bonne clé. Les systèmes d'antidémarrage sont également basés sur ce système, appliqué cette fois entre plusieurs ECU.

De même, les communications V2X requièrent un chiffrement et une signature forts des messages émis ou reçus par une voiture [MLLS13]. En effet, il est crucial de garantir l'authenticité, l'intégrité et la confidentialité de tels messages car

ceux-ci peuvent avoir un impact sur le comportement du véhicule qui en est le destinataire. De même, les communications entre véhicules étant nécessairement sans fil, la confidentialité des échanges doit être également assurée. Par conséquent, le système responsable des communications V2X sera nativement capable d'effectuer des opérations cryptographiques complexes.

L'extension de ce type de mécanismes à l'ensemble (ou à des sous-systèmes) du réseau interne permettrait ainsi de sécuriser les communications entre calculateurs d'une manière équivalente aux communications entre véhicules. En effet, l'ajout de méthodes sécurisées d'authentification peut permettre de garantir l'authenticité des messages, la possibilité de les signer garantit également l'intégrité et l'envoi de messages chiffrés permet la confidentialité. Cependant, l'implémentation de tels mécanismes dans une automobile doit tenir compte des contraintes suivantes :

1. Certains réseaux embarqués offrent des débits relativement faibles et possèdent déjà un fort taux d'encombrement. Or, il faut que l'utilisation de protocoles d'authentification ou l'ajout d'une signature aux messages ne provoque pas une trop forte augmentation du trafic, car les délais induits dans la réception d'un message nuiraient alors au bon fonctionnement du véhicule. En effet, un message CAN contient au plus 8 octets de données. Or une signature cryptographique de moins de 8 octets n'est plus considérée suffisante en termes de sécurité aujourd'hui. Par conséquent, une implémentation d'un système de signature des messages sur CAN implique nécessairement une fragmentation des données sur plusieurs trames, et donc une augmentation du trafic.
2. De plus, un ECU dispose d'une puissance de calcul limitée. La puissance de calcul requise pour les opérations cryptographiques ne doit pas impacter négativement le temps de traitement des autres tâches (contrainte de temps réel, cf. §1.1.1). Cependant, utiliser des algorithmes de chiffrement trop simplistes rendrait ces mécanismes caducs car un attaquant pourrait les casser très rapidement.
3. Il faut être capable de garantir une gestion sécurisée des clés de chiffrement, pour leur distribution ainsi que pour leur stockage (il faut envisager qu'un attaquant ait pu obtenir un accès physique au calculateur et possède du matériel permettant d'en lire le contenu). La mise en place d'une infrastructure à clés publiques (ou PKI, pour *Public Key Infrastructure*) permettant le renouvellement et la révocation de clés compromises s'avère ainsi nécessaire.
4. Enfin, la durée de vie d'une automobile peut être supérieure à la durée de vie des techniques de chiffrement qu'elle emploie. En effet, l'évolution constante du matériel informatique réduit le temps nécessaire pour casser un code, jusqu'à le rendre inefficace. Il faut alors pouvoir utiliser une clé plus longue, ou un algorithme de chiffrement différent.

La première contrainte sera en grande partie levée lorsque les réseaux embarqués utiliseront massivement des technologies plus récentes. Une trame FlexRay contient par exemple 254 octets de données, une trame Ethernet jusqu'à 1500. De

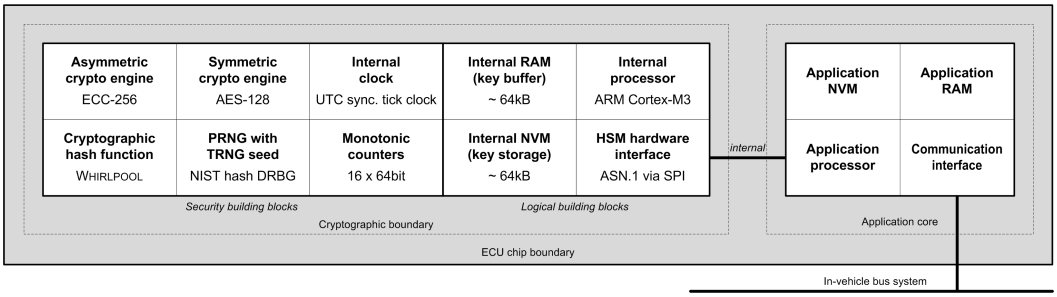


FIGURE 3.2 – Architecture globale d'un HSM (full)

nouvelles implémentation du protocole CAN pourraient également permettre de résoudre partiellement ce problème. Ainsi, CAN-FD [CAN12] permet d'avoir jusqu'à 64 octets de données dans une seule trame. De même, [VHSV11] et [GMVHV12] utilisent le protocole CAN+ [ZWT09] pour insérer les données nécessaires à l'authentification parmi le trafic CAN standard. En attendant, des méthodes de chiffrement peuvent être appliquées aux messages ne faisant pas l'objet de contraintes temporelles fortes ou étant déjà soumis à des fragmentations, comme les trames de diagnostic (cf. 1.2.7).

Concernant les contraintes 2 et 3, il est possible de délester le processeur d'un ECU des opérations cryptographiques, en le complétant avec un module dédié à la sécurité. Il existe désormais plusieurs implémentations de ce type. Nous pouvons par exemple citer une implémentation de *Trusted Platform Module* (TPM) [TCG11] pour l'automobile [6], ainsi que le *Hardware Security Module* (HSM) provenant du projet EVITA [HRS+09]. Nous utilisons ce dernier pour illustrer le principe dans la section 3.3.1.3. La conception et le déploiement de PKI par les constructeurs automobiles sont également en cours.

La dernière contrainte pourra être prise en compte grâce à des mécanismes de mise à jour des calculateurs *over the air* pour les algorithmes implantés logiciellement. Une implantation matérielle des algorithmes dans les ECU offre de meilleures performances mais devient plus complexe à maintenir. Dans ce cas, [HKD11] propose un recours au FPGA (*Field-Programmable Gate Array*) afin de réduire ce défaut.

3.3.1.3 Module de sécurité

Comme illustré en figure 3.2, le HSM de EVITA est constitué d'un processeur, de la mémoire volatile (RAM) et non-volatile (NVM), des blocs fonctionnels pour la sécurité et une interface avec le module applicatif (i.e. le cœur dédié aux opérations standards de l'ECU). Par ailleurs, afin de minimiser les coûts, il existe 3 niveaux de spécifications du HSM, *full*, *medium* et *light*, en fonction des besoins en termes de fonctionnalités de sécurité de l'ECU concerné.

6. `http://www.trustedcomputinggroup.org/resources/tcg_tpm_20_library_profile_for_automotivethin`

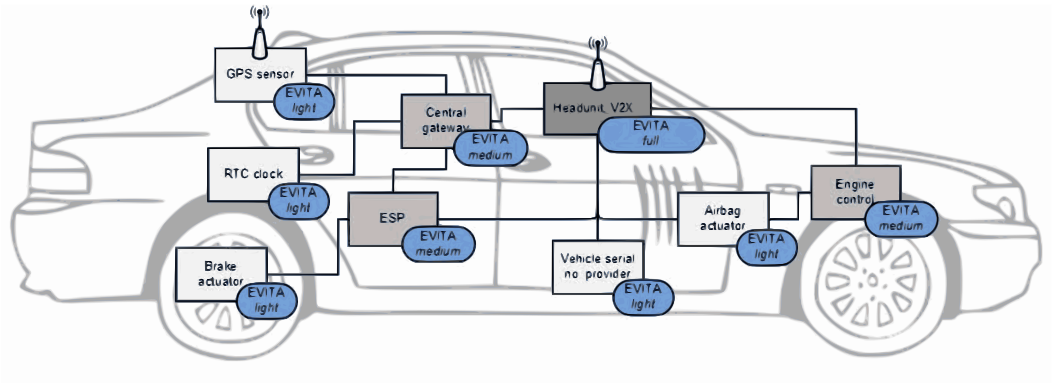


FIGURE 3.3 – Exemple de déploiement des différents modules HSM dans une voiture

- Le module *full* possède tous les blocs présents en figure 3.2. Il permet d'assurer la sécurisation des communications V2X (qui requièrent des algorithmes de cryptographie asymétrique performants) ainsi que des communications entre le réseau interne et les interfaces avec l'extérieur.
- Le module *medium* est dédié aux communications entre les ECU. Il possède un processeur moins puissant que le *full*, il ne conserve que l'accélération matérielle pour le chiffrement à clé symétrique mais est néanmoins capable de faire des opérations à clé asymétrique logiciellement (ce qui suffit s'il n'y a pas de contrainte de temps trop forte associées aux messages utilisant ce chiffrement).
- Le module *light* est à son tour une version allégée du module *medium* et ne permet de faire que des opérations à clé symétrique. Il est conçu pour sécuriser les communications entre un ECU et des capteurs.

La figure 3.3 illustre une disposition suggérée des différents modules dans une automobile : le module *full* est ainsi uniquement utilisé pour l'ECU responsable des communications avec l'extérieur, des modules *medium* sont présents dans les ECU contrôlant des sous-systèmes de la voiture et les modules *light* sont utilisés pour les capteurs et actionneurs eux-mêmes.

Cependant, les améliorations apportées par un HSM ne permettent pas nécessairement d'employer n'importe quel algorithme de chiffrement selon la situation (et l'impact sur le prix du système). En effet, comme mentionné précédemment, certains systèmes sont soumis à des contraintes de temps réels plus fortes que d'autres, et doivent donc rendre leur service en un temps borné. Dans ce genre de situation, l'ajout d'un chiffrement symétrique permet de réduire l'impact sur le temps d'exécution, voire un coût inférieur, mais la gestion d'un grand nombre de clés secrètes implique des contraintes plus fortes quant à leur stockage (à ce sujet, [SRW+11] propose une architecture de sécurité pour fournir des protocoles de partage de clé secrète sécurisés et de signature de messages via un ECU dédié à la gestion des clés). Des approches combinant les deux modes de chiffrement sont à envisager, où les messages sont chiffrés grâce à des clés symétriques préalablement échangées

grâce à un protocole de partage basé sur un chiffrement asymétrique, d'une manière similaire à ce qui se fait par exemple dans SSL ou IPSEC.

3.3.2 Sécurité des données

Nous allons maintenant présenter des mécanismes visant à assurer l'intégrité des données contenues dans les ECU. Le but est ici d'empêcher ou de limiter les possibilités d'un attaquant cherchant à utiliser les données stockées dans un ECU, que ce soit pour en prendre le contrôle en exploitant une vulnérabilité logicielle ou récupérer des informations contenues dans celui-ci (code du logiciel, données personnelles ...). Ces mécanismes servent ainsi principalement à se prémunir d'attaques ciblant le logiciel embarqué, généralement afin d'empêcher toute compromission durable du système.

3.3.2.1 Signature du code

En plus de permettre une sécurisation des communications, l'ajout de modules de cryptographie aux ECU apporte aussi la possibilité de vérifier l'authenticité de code, que ce soit lors du démarrage de l'ECU, d'une manière similaire aux processus de *secure boot* [AFS97] existant par ailleurs, ou suite au téléchargement d'une mise à jour. L'intérêt de cette démarche est d'empêcher l'exécution par le calculateur de code n'ayant pas été signé par une source reconnue (c'est à dire un développeur approuvé par le constructeur).

La mise en place d'une base de confiance dans un véhicule peut ainsi se faire grâce aux modules de sécurité mentionnés en 3.3.1.2.

3.3.2.2 Virtualisation

Désormais une pratique courante dans l'informatique traditionnelle, la virtualisation permet d'isoler entre eux des composants logiciels partageant un même matériel. Concrètement, un module logiciel nommé hyperviseur accueille plusieurs systèmes d'exploitation dits invités, pas nécessairement identiques, et est chargé d'assurer la répartition des ressources matérielles permettant l'exécution en parallèle de ces invités. En présentant aux invités une abstraction de la couche matérielle, l'hyperviseur est ainsi capable d'allouer une portion déterminée des ressources matérielles à chaque système invité et d'en restreindre l'accès aux autres.

Dans le cadre de l'automobile, nous pouvons citer le projet européen Oversee [GHR+09] dont un des objectifs est d'assurer l'intégrité du système multimédia d'une voiture par la virtualisation (basée sur l'hyperviseur XtratuM [7]). La figure 3.4 illustre cette solution : des applications de différents niveaux de criticité, s'exécutent dans des OS invités (potentiellement différents) au dessus d'un hyperviseur chargé de sécuriser les communications entre les différents systèmes invités. On voit par exemple ici que les communications entre invités ne peuvent se faire que dans un

7. `http://www.xtratum.org/`

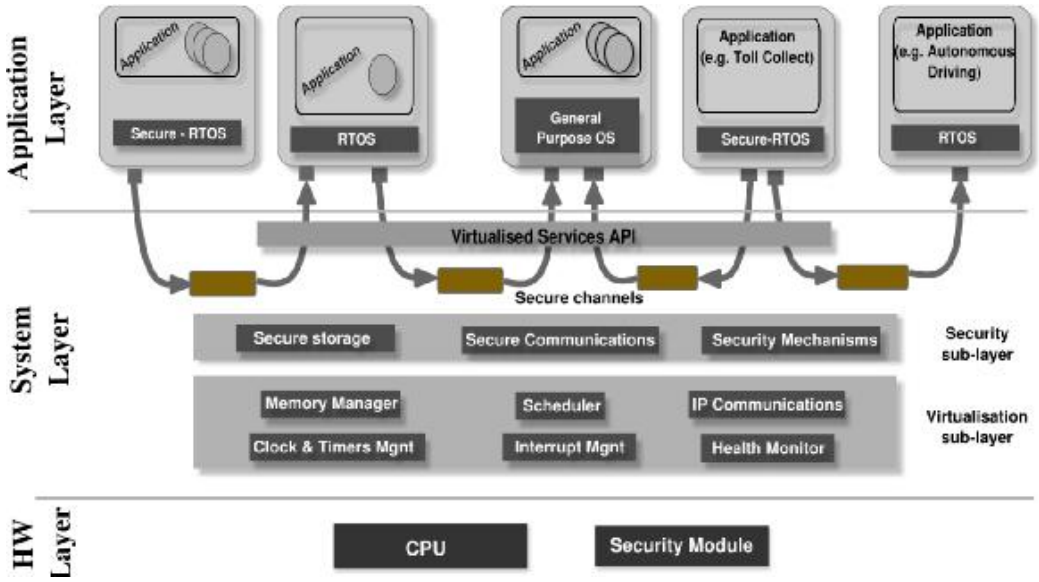


FIGURE 3.4 – Architecture virtualisée proposée dans le projet Oversee [GHR+09]

sens : du système le plus critique vers un système de criticité inférieure (sens des flèches sur la figure). Ainsi, si un attaquant réussit à exploiter une vulnérabilité d'un protocole de communication sans fil, tel que présenté en 2.3, son pouvoir de nuisance sera restreint : il ne pourra pas contaminer les fonctionnalités les plus critiques de l'ECU concerné (en supposant bien sûr que l'hyperviseur parvient à garantir une isolation stricte).

Au delà d'un renfort de la sécurité d'un ECU existant, la virtualisation offre d'autres avantages. Tout d'abord, comme il devient possible de regrouper de manière sécurisée des tâches de criticités différentes au sein d'un unique ECU, cela peut permettre de réduire les coûts matériels sans impacter négativement la sécurité du système. De plus, le recours à un hyperviseur peut aussi permettre d'effectuer une surveillance en profondeur d'un système invité « depuis l'extérieur », sans altération de celui-ci.

En revanche, il conviendra de s'assurer que l'hyperviseur en lui-même ne comporte pas de vulnérabilités, car cela ne reviendrait alors qu'à ouvrir de nouvelles possibilités d'attaque (certes plus complexes). À ce sujet, le recours à des hyperviseurs de petite taille permet d'effectuer des vérifications en profondeur de leur code et ainsi de garantir leur sécurité.

3.3.2.3 Contrôle d'accès

Nous désignons le contrôle d'accès comme l'aptitude d'un système à contrôler les capacités des *sujets* (dans les cas qui nous intéressent, des entités informatiques actives du système telles que des programmes ou des processus) à accéder aux *objets* (les ressources et données) de celui-ci. Une politique de contrôle d'accès définit ainsi différent niveaux de privilèges qui permettront de décider si un sujet a le droit

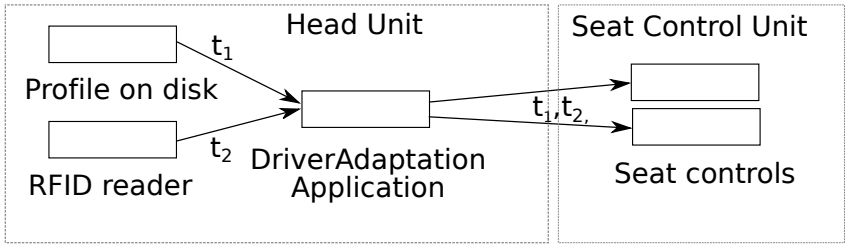


FIGURE 3.5 – Marquage des données dans [SR12]

d'accéder (utiliser une ressource, lire ou écrire des données) à un objet particulier.

Dans le cadre de l'automobile, de telles politiques ne sont pour l'instant pas déployées (à notre connaissance) sur le réseau embarqué de véhicules actuellement en circulation. En revanche, nous pouvons trouver des travaux sur le sujet dont l'objectif est d'intégrer de telles techniques dans les architectures existantes. Ainsi, [SR12] et [BSWE13] ont proposé des mécanismes basés sur le contrôle des flux d'informations (*Information Flow Monitoring*), dans lesquels certaines données et ressources possèdent un marqueur qui permet de suivre la propagation de ces données (qui les a générées, qui les a utilisées). La figure 3.5 illustre ce principe. Dans cette illustration, le modèle de voiture est capable de sauvegarder dans un fichier les profils de plusieurs conducteurs contenant l'ensemble des réglages (position du fauteuil, inclinaisons des rétroviseurs ...) et de les restaurer lorsque ceux-ci sont au volant. Pour ce faire, les différents conducteurs sont identifiés grâce à leurs clés (chaque clé possédant un identifiant unique). Ainsi, lorsqu'un conducteur ouvre la voiture, au sein de l'ECU *Head Unit*, l'application responsable du réglage des fauteuils doit accéder à l'identifiant de la clé, fourni par un lecteur RFID ainsi qu'à l'ensemble des profils sauvegardés. L'application doit ensuite envoyer des instructions à deux applications de l'ECU responsable des réglages du fauteuil pour qu'il s'adapte au profil du conducteur détecté. Les données sont marquées dès leur origine (marqueur t1 pour le profil du conducteur, t2 pour l'identifiant de la clé). Les nouvelles données générées (à destination de l'ECU de contrôle du fauteuil) suite à leur utilisation par l'application sont alors marquées avec t1 et t2. Dans [BSWE13], il y a par exemple 4 valeurs de marqueurs, de 0 à 3, correspondant à 4 niveaux de confidentialité (3 représentant le niveau le plus fort). Les politiques de contrôle d'accès vont ainsi déterminer à partir des marqueurs présents sur les données si les requêtes émises par des sujets souhaitant y accéder sont légitimes. Si ce n'est pas le cas, le système en refuse l'accès au sujet, préservant la confidentialité de ces données. Le scénario de la figure 3.5 implique ainsi quatre règles : deux pour autoriser la récupération des données provenant des deux sources, et deux pour les deux flux de données sortant de l'application à destination des contrôles du fauteuil.

Une autre approche a été proposée dans [SPS+09], avec l'introduction d'un système de gestion de données (DMS, pour *Data Management System*). Le principe est qu'au lieu de laisser les ECU gérer les échanges d'information entre eux, les données sont regroupées dans des nœuds spécifiques du réseau. Dès lors, des mécanismes

de contrôle d'accès peuvent être facilement inclus dans le DMS pour préserver la confidentialité des données.

3.3.3 Gestion du réseau

Au vu des problématiques de sécurité posées par l'arrivée de systèmes de (télé)communications dans les véhicules, une solution en apparence évidente serait de séparer entièrement le (ou les) réseau(x) contenant des interfaces de communication avec l'extérieur (typiquement, le réseau multimédia) des réseaux critiques, responsables du bon fonctionnement du véhicule, voire de regrouper les ECU sur des réseaux isolés selon leur niveau de criticité (voir 1.4.1). Ainsi, une attaque à distance lancée contre le véhicule n'aboutirait au pire qu'en la compromission des équipements multimédia. Cependant, une telle architecture se heurte hélas à plusieurs contraintes sur les véhicules modernes. En effet, certains cas d'utilisation souhaités par les constructeurs nécessitent une collaboration entre les équipements multimédia et le reste du véhicule. Par exemple, l'interface multimédia peut servir à mémoriser et rappeler différents profils de conducteurs, et donc envoyer des instructions au reste du véhicule. De même, les communications car2car pourront à terme permettre au véhicule de réagir de lui-même en fonction des informations qui lui seront transmises, nécessitant de fait une liaison entre le module de communications et les commandes du véhicule.

Nous présentons dans la suite les mécanismes de sécurité qui concernent le réseau, en particulier, les moyens se basant sur la conception du réseau pour se protéger d'un attaquant. Ceux-ci concernent la structure du réseau et la présence de pare-feux aux interfaces entre sous-réseaux. Comme son nom l'indique, cette catégorie vise à prévenir tout type d'attaques interagissant avec le réseau.

3.3.3.1 Architecture

Une séparation complète n'étant donc pas envisageable, la prise en compte des problématiques de sécurité lors de la conception de l'architecture réseau est nécessaire. Il est en effet possible de diminuer la sévérité d'une attaque contre un véhicule en jouant sur l'architecture du réseau et la disposition des ECU dans celle-ci. L'impact d'une attaque lancée contre le réseau embarqué variera notamment en fonction du nombre et la variété de points d'entrée disponibles, de la présence de systèmes pouvant contrôler automatiquement des éléments critiques du véhicule, ainsi que de la topologie du réseau, qui contribuent à augmenter la difficulté qu'aura un attaquant à atteindre sa cible (si celle-ci diffère d'un point d'entrée). L'étude menée par Miller et Valasek [MV14] illustre bien ce fait en tentant d'évaluer la difficulté qu'un attaquant aura à contrôler des véhicules en fonction de la topologie de leur réseau. Considérons les deux exemples de la figure 3.6, tirés de cette étude. Dans l'architecture présentée à gauche, le système multimédia (« radio ») regroupe de nombreux points d'entrée pour une attaque (téléphonie, bluetooth, internet). Or cet ECU est directement connecté (la ligne colorée en rouge) au bus contenant les

systèmes critiques. Dès lors, en cas de compromission du multimédia, un attaquant pourra communiquer directement avec les ECU contrôlant le véhicule, mettant en danger ses occupants. En revanche, dans la figure de droite, le réseau est bien plus fragmenté, et une passerelle centrale (encadrée en vert) sert d'interface entre chacun des sous-réseaux. Dans ce cas, il sera plus difficile à un attaquant s'étant introduit par un des points d'entrée d'accéder aux éléments critiques du réseau.

3.3.3.2 Pare-feux

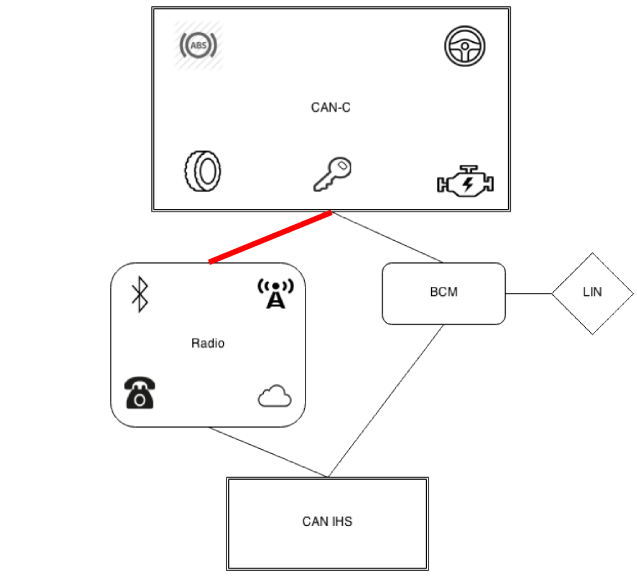
L'interconnexion entre équipements de communication et modules de contrôle du véhicule étant requise, le renforcement de la sécurité au niveau des passerelles est un point crucial. Ces passerelles sont généralement capables d'effectuer un minimum de filtrage : elles ne relayeront pas sur un bus A des données provenant du bus B mais n'étant destinées à aucun nœud de A. En soi, elles agissent donc déjà comme une sorte de pare-feu rudimentaire.

Cependant, un mécanisme aussi simple peut être contourné [HKD09] et n'est pas capable de faire de suivi de connexion dans le cadre de protocoles plus avancés (comme par exemple pour les protocoles de diagnostic). Pour pallier ces manquements, il est possible d'étendre les capacités des passerelles, les rapprochant alors des systèmes de pare-feux existants dans l'informatique traditionnelle. Un pare-feu adapté à l'automobile peut ainsi faire du suivi de connexion ou encore d'inspecter le contenu des trames afin de déterminer s'il est légitime de relayer l'information [Ari12]. Bien évidemment, l'efficacité des pare-feux est liée à la topologie du réseau et à leur emplacement dans celui-ci. Opérant aux interfaces entre les réseaux, ils joueront un rôle d'autant plus important qu'il y aura de branches partant de leur emplacement, l'idéal étant un pare-feu au centre d'un réseau en étoile.

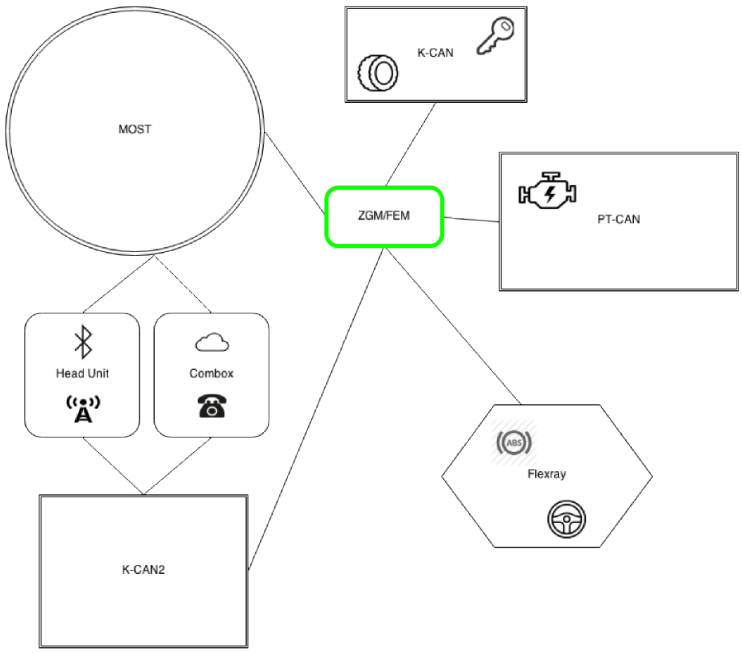
3.4 Techniques de sécurité réactives

Par opposition aux mécanismes préventifs, les mécanismes réactifs entrent en jeu une fois que l'attaquant a réussi à pénétrer le système. Nous les avons regroupés en fonction de leurs objectifs :

- Les systèmes de détection d'intrusion, qui comme leur nom l'indique, cherchent à détecter la présence d'un attaquant dans le système, et éventuellement à contrer ses attaques.
- Les analyses *forensics*, procédés consistant à analyser suite à un incident les données enregistrées par le système afin d'identifier de possibles attaques, et ainsi fournir les moyens d'anticiper de futures tentatives similaires.
- Les pots de miel, qui servent à leurrer des attaquants, que ce soit pour les détourner de leur cible principale ou être capable d'observer leur comportement dans un environnement contrôlé.



(a)



(b)

FIGURE 3.6 – Exemples de topologies de réseaux automobiles tirés de [MV14]

3.4.1 Détection d'intrusion

La mise en place de mesures préventives comme celles décrites précédemment, est une condition nécessaire à la sécurité d'un système, elle n'est en revanche pas suffisante. En effet, tout mécanisme a ses limites et si un attaquant parvient à contourner les protections mises en place et s'introduit dans le système, celles-ci ne seront pas capables de détecter sa présence, ni d'identifier ses actions.

L'objectif de la détection d'intrusion est ainsi de prendre le relais sur les systèmes de sécurité préventifs, afin de savoir quels ont été les actes de l'attaquant suite à son intrusion. Cela peut permettre de comprendre comment l'intrusion a pu avoir lieu, d'identifier les cibles de l'attaque et de réparer les éventuels dégâts causés afin de se prémunir plus efficacement contre de futures attaques.

3.4.1.1 Définitions

Un système de détection d'intrusion (ou IDS, pour *Intrusion Detection System*) est installé au cœur du système à surveiller, et non à ses entrées (contrairement à un pare-feu, par exemple). Désormais courants dans l'informatique moderne, ceux-ci couvrent une grande variété [LKS05] de situations. Pour exprimer cette diversité, nous reprenons ici la classification présentée dans [Mic03], illustrée par la figure 3.7.

Les IDS peuvent ainsi être caractérisés de plusieurs façons :

1. En fonction de la source des données utilisées pour la détection d'intrusion. Historiquement, on distinguait entre les données fournies par un système d'exploitation et celles provenant du trafic réseau. On parle ainsi d'IDS *host-based*, dans le premier cas et *network-based* dans le second. Il existe également des IDS utilisant des données provenant d'applications ou encore d'autres IDS. Dans ce dernier cas, l'idée est de corréler plusieurs alertes pour identifier des attaques de plus haut niveau.
2. En fonction de leur méthode de détection. Historiquement, une distinction est faite entre les IDS utilisant une approche par scénarios et ceux utilisant une approche comportementale.
   (a) Dans le cas d'une approche par scénarios, l'IDS possède une base de scénarios décrivant des attaques. Une alerte est levée quand l'IDS reconnaît un des ces scénarios au cours de son observation. Les attaques à détecter étant préalablement connues, une approche par scénarios permet de déterminer précisément l'attaque qui a été détectée ainsi que de limiter fortement l'apparition de faux positifs (une alerte est levée alors qu'il n'y a pas d'attaque). La contrepartie est que toute attaque inconnue ne pourra pas être détectée (faux négatif) tant que le système n'aura pas été mis à jour.
   (b) Une approche comportementale consiste à modéliser le comportement attendu du système à un instant donné. Les données collectées sont ensuite confrontées à ce modèle, toute déviation par rapport à celui-ci étant considérée comme suspecte. Contrairement à l'approche par scénarios,

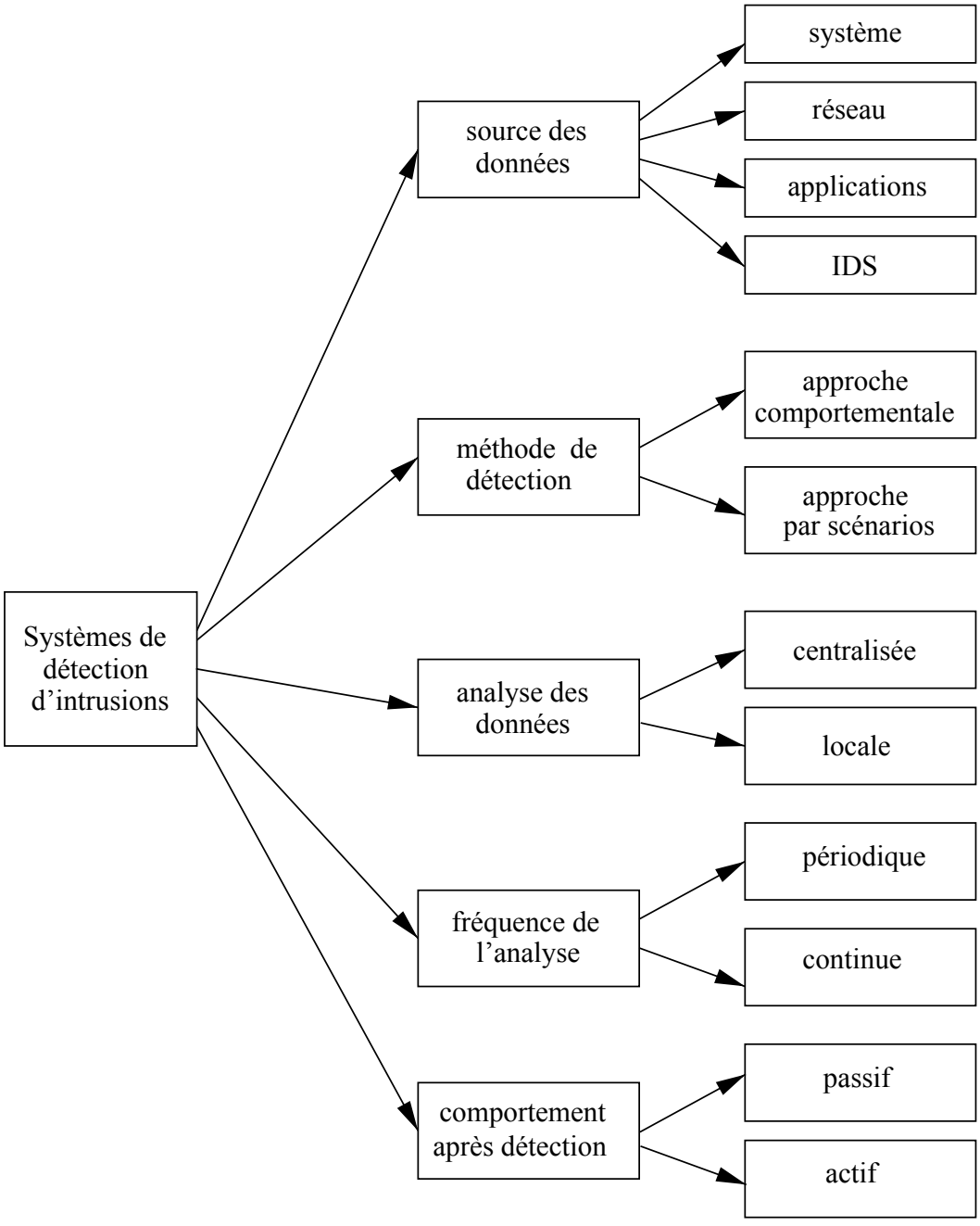


FIGURE 3.7 – Caractéristiques des systèmes de détection d'intrusion [Mic03]

cette approche peut permettre de détecter des attaques jusqu'alors inconnues. Cependant, il n'est pas toujours aisé d'obtenir un modèle exhaustif du système à surveiller, ce qui risque de provoquer de faux positifs.

3. En fonction de l'emplacement où se fait l'analyse des données. Celle-ci peut être locale (elle est effectuée au plus près de la source de données, pas de croisement d'informations entre sources distinctes) ou centralisée (les données de différentes sources sont rapatriées vers une unique machine et peuvent donc être corrélées).
4. En fonction de la fréquence de l'analyse. Celle-ci peut être effectuée périodiquement, à la recherche d'intrusions passées, ou continue, pour une surveillance en temps réel.
5. En fonction de leur comportement après détection. Un IDS passif se contente de lever une alerte en cas d'intrusion détectée, alors qu'un système actif tentera de stopper automatiquement une attaque en cours. On parlera alors d'IPS (*Intrusion Prevention System*, système de prévention d'intrusion).

3.4.1.2 Applications dans l'automobile

Un environnement automobile ne posant pas les mêmes contraintes qu'un réseau informatique traditionnel, l'adaptation directe de systèmes existants par ailleurs n'est pas envisageable. Par exemple, on ne peut pas considérer qu'une voiture soit capable de télécharger à tout moment une mise à jour logicielle, ce qui compromettra l'efficacité d'un IDS dont la base de scénarios n'est pas à jour (rappelons que la durée de vie d'une voiture avoisine les 20 ans, ce qui implique également un suivi de longue durée). Par ailleurs, les méthodes d'alerte et/ou de réaction doivent également être adaptées à ce contexte pour ne pas mettre en danger inutilement les passagers. [HKD08] propose par exemple trois moyens pour alerter le conducteur en fonction de la gravité de l'intrusion détectée : visuel, auditif puis haptique.

À l'heure actuelle, l'application de la détection d'intrusion aux systèmes embarqués dans l'automobile en est encore à ses débuts. Ainsi, les exemples présentés ci-après, tirés de la littérature, constituent des premières pistes de réflexion et sont au mieux des preuves de concept [8]. Larson et al. [LNJ08] proposent une approche où chaque ECU possède un capteur qui observe les interactions de ce dernier avec le réseau (les messages émis mais aussi les messages consommés). La détection d'intrusion repose sur un ensemble de règles de sécurité basées sur les spécifications du protocole réseau et de l'ECU hôte. La détection d'intrusion se fait indépendamment dans chaque ECU. Les ECU passerelles, qui possèdent un capteur par bus auquel ils sont rattachés, peuvent corréler les informations obtenues via ceux-ci et ainsi détecter plus d'attaques.

[MGF10] décrit 8 capteurs aux rôles distincts (voir tableau 3.1), chargés de vérifier que les trames circulant sur le réseau sont conformes à un ensemble de règles

8. Des solutions commerciales de détection d'intrusion commencent à apparaître (voir par exemple `http://www.tower-sec.com`, mais aucune information concrète n'est donnée sur leur fonctionnement.

préalablement définies. La détection d'intrusion se base sur une fenêtre glissante pendant laquelle les alertes remontées par chaque capteur, symbolisées par une valeur correspondant à l'importance attribuée à ceux-ci, sont sommées. Trois niveaux d'incidents sont définis, (important, critique, sévère), correspondant chacun à un seuil devant être atteint par cette somme.

Capteur	Description
Formalité	Bon format : taille du message, de l'en-tête, de chaque champ, checksum correct, etc.
Emplacement	Le message circule bien sur un bus autorisé
Étendue des données	Les données transmises sont comprises dans un certain intervalle
Fréquence	Le timing des messages est correct
Corrélation	La corrélation des messages à travers différents bus correspond aux spécifications
Protocole	Les protocoles de type challenge-réponse se déroulent bien dans le bon ordre, au bon moment, etc.
Plausibilité	Le contenu des données est plausible, il n'y a pas de conflit avec les valeurs précédemment observées
Cohérence	Les données provenant de sources redondantes sont cohérentes entre elles

Table 3.1 – Liste des capteurs définis par [MGF10]

[MA11] définit une entropie du bus CAN, qui est calculée en observant le trafic lors d'une activité « normale » de référence sur un bus CAN. Une alerte est alors levée lorsque des déviations de l'entropie sont constatées comparativement à des valeurs de référence.

De même, le système présenté dans [MV14], connecté au port OBD, observe le trafic réseau pendant une phase d'apprentissage. Puis, si une anomalie est détectée, il court-circuite le bus afin d'empêcher toute émission future de message.

Dans [MHT+12], la solution présentée repose sur la surveillance du trafic réseau par chacun des ECU présents. Lorsqu'un ECU observe un message circulant sur le bus dont il est censé être l'émetteur (en se basant sur l'identifiant de ce message) alors que ce n'est pas le cas, cet ECU envoie immédiatement une trame d'alerte sur le bus afin d'écraser le message (jugé illicite) en cours d'émission.

Le tableau 3.2 résume les exemples précédents, présentés selon la classification énoncée précédemment.

Solution	source des données	méthode de détection	emplacement	fréquence	comportement
[LNJ08]	système	scénarios	local	continue	passif
[MGF10]	réseau/IDS	comportementale	distribué	continue	passif
[MA11]	réseau	comportementale	local	continue	passif
[MHT+12]	réseau	scénarios	local	continue	actif
[MV14]	réseau	comportementale	local	continue	actif

Table 3.2 – Solutions de détection d'intrusion dans l'automobile

3.4.1.3 Détection bas-niveau

Une autre approche, proposée dans [MG14] décrit une méthode pour identifier la source d'un message en se basant sur les caractéristiques physiques du signal émis. L'idée est de se placer non pas au niveau du protocole mais sur la couche physique afin de définir une empreinte de chaque nœud du réseau. Les spécifications CAN laissant une grande liberté dans l'implémentation de la couche physique, il devient alors possible de distinguer les produits provenant de différents fournisseurs. Les composants électroniques eux-mêmes possédant chacun des caractéristiques uniques, il peut même être possible d'opérer une distinction entre des émetteurs provenant du même fournisseur. S'il semble illusoire d'utiliser une telle méthode pour proposer une méthode d'authentification en raison des marges d'erreurs parfois non négligeables, son application dans le domaine de la détection d'intrusion parait envisageable en complément d'autres approches, afin de détecter le rajout ou le remplacement d'un nœud du réseau par un attaquant.

3.4.2 Pots de miel

En informatique, un pot de miel (voir par exemple [SNAK12]) consiste en un système, réel ou simulé, généralement peu ou pas protégé contre les intrusions, conçu afin d'observer des attaquants. Isolé du système à protéger et surveillé en permanence, un pot de miel agit de fait comme un leurre, permettant à celui qui l'a mis en place de détourner des attaquants de la cible réelle et d'observer les stratégies qu'ils mettent en place lors d'une attaque. Afin d'être le plus efficace possible (c'est à dire afin qu'un attaquant se doute le plus tard possible qu'il n'est pas dans le système réel), un pot de miel se doit d'interagir avec l'attaquant de la façon la plus réaliste possible. En contre partie, il convient de s'assurer qu'un plus grand nombre d'interactions offertes à l'attaquant au sein du pot de miel ne débouche pas sur une véritable attaque (par exemple si l'attaquant utilise alors le pot de miel pour attaquer une autre machine). Une distinction est ainsi faite entre pots de miel haute interaction, offrant la plus grande similarité avec un système réel, et pots de miel basse interaction, offrant plus de garanties de sécurité au détriment de l'interactivité.

L'automobile connectée n'en étant encore qu'à ses débuts, on ne possède à l'heure actuelle que très peu d'informations concrètes concernant d'éventuels attaquants. Une automobile équipée d'un pot de miel se déplaçant dans une zone donnée pourrait ainsi permettre d'en savoir plus sur la présence et les motivations d'éventuels attaquants avant que des attaques avérées ne se produisent contre des automobiles. Une première approche d'adaptation de pot de miel à l'automobile a été décrite dans [VNLJ08]. Comme pour des raisons de sécurité évidentes, il est impensable d'utiliser directement le réseau embarqué d'un véhicule réel pour servir de pot de miel dans un tel scénario, l'approche (voir figure 3.8) consiste en une voiture réelle possédant une interface de communications sans fil (*wireless gateway*) uniquement connectée à un pot de miel simulant le réseau automobile, en utilisant des

outils du type de CANoe. Ce véhicule est alors censé circuler normalement pendant que le réseau simulé enregistre toute tentative d'attaque.

Tenant compte de la nécessité d'un compromis entre réalisme de la simulation et sécurité du système, les auteurs décrivent trois modèles proposant des degrés de réalisme différents.

1. Dans le premier cas, les instructions correspondant aux actions du conducteur et aux données de l'environnement sont pré-enregistrées et jouées lors de l'expérimentation. Celles-ci sont donc entièrement décorrélées de ce que fait réellement le véhicule. N'offrant aucune interaction avec le véhicule réel, ce modèle est le plus sûr et le plus aisé à implémenter, au détriment d'un réalisme moindre.
2. Dans le deuxième cas, les actions du conducteur sont générées via un modèle comportemental, réagissant ainsi en direct aux stimuli lus sur le réseau simulé. Bien plus complexe à mettre en place, ce modèle présente un réalisme accru pour le même niveau de sécurité que le précédent. En revanche, un attaquant pourra peut être se rendre compte que les informations qu'il lit ne correspondent pas aux actions réelles du véhicule attaqué.
3. Dans le dernier cas, le réseau simulé reçoit les véritables données correspondant aux actions du conducteur ainsi qu'à l'environnement. Dans ce cas, les informations vues par l'attaquant correspondent vraiment à ce qui reçu par le véhicule réel (du moins tant que l'attaquant ne cherche pas à perturber son fonctionnement). En revanche, des risques de sécurité apparaissent puisque des interfaces relayant les données reçues par le réseau réel vers le réseau virtuel deviennent nécessaires.

À l'heure actuelle, il n'existe pas à notre connaissance d'implémentation concrète de pots de miel dans une automobile. En l'état, des recherches supplémentaires visant à assurer un déploiement sécurisé de pots de miel automobiles de type haute interaction nous semblent nécessaires si de tels dispositifs sont envisagés dans le futur.

3.4.3 Analyses forensics

Les analyses légales sont effectuées après un incident afin d'identifier précisément les circonstances de celui-ci. Avec l'émergence d'attaques informatiques lancées contre des automobiles, le risque qu'un incident s'avère lié à une attaque contre des calculateurs embarqués ne sera plus nul, ce qui entraînera un besoin croissant pour de telles analyses. Il est donc important de s'assurer que les données nécessaires pour conduire ces analyses puissent-être collectées et stockées de manière sécurisée car elles pourront par exemple permettre de différencier une attaque d'un simple accident. En effet, une telle collecte de données deviendrait inutile dans le cadre d'une attaque informatique si l'attaquant réussit à accéder à celles-ci pour les modifier. À l'heure actuelle, le réseau embarqué possède déjà un système de rapport d'incidents, via les processus de *self-diagnostic*. Cependant, ceux-ci, prévus

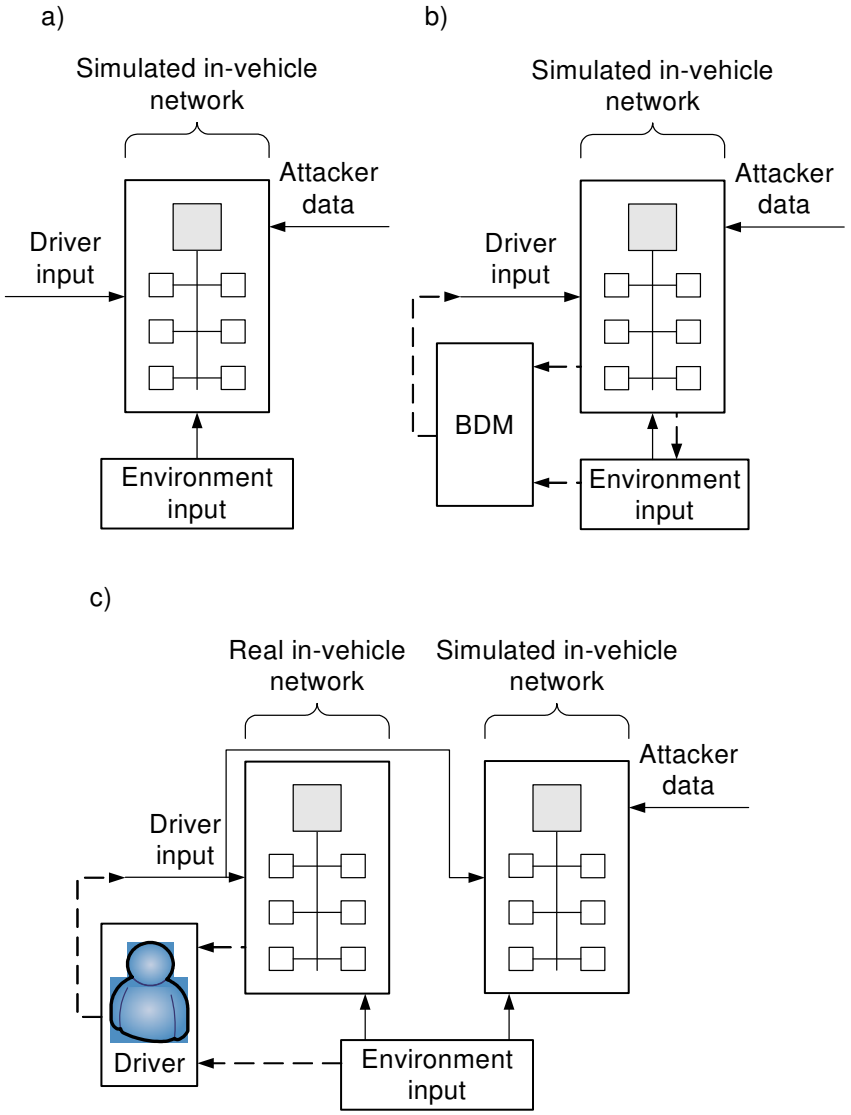


FIGURE 3.8 – Modèle de déploiement de pot de miel dans une automobile proposé par [VNLJ08]

pour aider un réparateur à localiser les sources d'une défaillance, se concentrent sur les événements relatifs à la *safety* et, en l'état, ne suffisent pas nécessairement à identifier les traces d'une attaque informatique. À cet effet, [HKD11] énonce 8 types de données pouvant être collectées dans une automobile actuelle, correspondant à 8 niveaux d'abstraction, qui peuvent se révéler utiles lors d'analyses :

1. des données matérielles (*hardware data*), peu ou pas influencées par les applications logicielles : numéros de série des composants ou encore données stockées en ROM (*Read Only Memory.*
2. des données brutes (*raw data*), c'est-à-dire des séquences de bits non traitées provenant du système : copies de la mémoire volatile ou encore flux de données brutes observé sur le CAN.
3. des métadonnées (*data about data*), comme par exemple l'horodatage des trames.
4. des données de configuration (*configuration data*) : ce sont des données qui peuvent être changées par le système d'exploitation ou les applications, qui influent sur le comportement du système mais pas sur les communications. L'exemple donné est le facteur permettant de déterminer la vitesse du véhicule en fonction du périmètre de la roue.
5. des données des protocoles de communication (*communication protocol data*), qui influent sur le comportement des communications. Ce sont par exemple les identifiants CAN, qui déterminent la priorité d'une trame.
6. des données de processus (*process data*), qui sont les données concernant un processus en cours d'exécution : son statut, sa priorité...
7. des données de session (*session data*) : des données collectées au cours d'une session. Dans le cas d'une automobile, ce sont typiquement les messages échangés lors d'une session de diagnostic.
8. des données utilisateur (*user data*) les données lues ou modifiées par l'utilisateur, par exemple des fichiers audio ou vidéo utilisés par le système multimédia.

En fonction des événements que l'on souhaite pouvoir identifier, il est alors suggéré d'étendre les capacités d'enregistrement d'incidents d'un véhicule en y incluant des mécanismes permettant de collecter plus efficacement certains de ces types de données. Cependant, si certaines données peuvent être collectées via un équipement dédié sur le réseau (approche centralisée), d'autre requerront une modification de chaque ECU (approche distribuée), nécessairement plus complexe et coûteuse. Il est donc important de définir précisément les attaques que l'on souhaite pouvoir mettre en évidence afin de pouvoir déterminer les types de données que l'on souhaite capturer pour pouvoir limiter les modifications à apporter au système et donc le surcoût entraîné par de telles méthodes.

Par ailleurs, un déploiement de tels dispositifs d'enregistrement doit également tenir compte des problématiques liées au respect de la vie privée des utilisateurs. En effet, certaines des données pouvant s'avérer utiles dans le cadre d'analyses *forensics*

peuvent également relever de la vie privée du conducteur. Leur stockage doit donc *a minima* se conformer à la législation en vigueur.

3.5 Conclusion

Nos travaux concernant les problématiques de sécurité dans les réseaux automobiles embarqués, nous avons présenté dans ce chapitre les différents mécanismes de sécurité informatique existants ou envisagés pour protéger les réseaux automobiles embarqués.

Du point de vue de l'informatique embarquée, une voiture moderne constitue un système complexe. Afin d'en proposer une sécurisation véritablement efficace, une stratégie de sécurité, prenant en compte tous les aspects de cette complexité doit être mise en place. Nous avons ainsi vu que les constructeurs automobiles ont désormais pris conscience des enjeux que représente la sécurité informatique pour les réseaux automobiles (embarqués et débarqués), comme en témoignent par exemple les divers projets européens que nous avons présentés. De plus, une automobile étant un système (de plus en plus) complexe, la mise en place de politiques de sécurité nécessite une vision d'ensemble du (ou des) système(s) concerné(s) pour être véritablement efficace. À cet effet, nous avons présenté le concept de défense en profondeur, qui est une tendance guidant actuellement le développement et l'intégration de mécanismes de sécurité pour les automobiles.

Nous avons ensuite fait un tour d'horizon des mécanismes de sécurité, selon le rôle qu'ils jouent dans la mise en place d'une stratégie de défense.

- Tout d'abord, les mécanismes dits préventifs, dont le but est d'empêcher un attaquant de pénétrer en certains points du système. Ceux-ci opèrent aussi bien à l'échelle des protocoles de communication, des données contenues dans les calculateurs, ou encore de l'architecture du réseau.
- Ensuite, les mécanismes qui entrent en jeu dans les cas où un attaquant a réussi à déjouer ou contourner les mesures préventives mises en place. Ceux-ci visent alors à détecter sa présence, à contrer ou à dévier ses attaques, mais peuvent également servir à en apprendre plus sur le comportement de tels attaquants, afin de renforcer la sécurité du système (ou des systèmes suivants) par la suite.

Cependant, les problématiques de sécurité étant apparues relativement récemment avec l'émergence de la voiture en tant qu'objet connecté, les solutions existantes actuellement sont encore amenées à évoluer dans les années à venir, guidées par les axes de développement définis par les constructeurs. De plus, comme vu dans le chapitre 1, la conception d'une automobile impose diverses contraintes aux équipements embarqués. Les solutions de sécurité déployées dans les automobiles futures devront donc trouver le compromis idéal entre le niveau de sécurité maximal et (notamment) le coût de déploiement.

Enfin, il nous semble important de mentionner que certains paramètres parfois ignorés peuvent permettre de passer outre les protections mises en place par un

FIGURE 3.9 – Ingénierie sociale utilisant le lecteur multimédia

dispositif de sécurité jugé pourtant idéal par ailleurs. Ainsi, l'impact du facteur humain n'est jamais à négliger lors de la conception d'un système : un utilisateur peut être victime d'ingénierie sociale visant à lui faire effectuer une manipulation dangereuse pour lui ou pour le système avec lequel il interagit (on pensera ainsi aux nombreuses tentatives d'hameçonnage reçues par courriel). [HKD11] illustre bien ce fait avec un fichier MP3 dont les métadonnées ont été éditées pour contenir un (faux) message d'alerte (figure 3.9), au lieu du titre du morceau et de l'interprète, qui s'affichera sur le lecteur multimédia durant sa lecture. Cette « attaque » ne viole ainsi aucune règle de sécurité, mais peut néanmoins fonctionner contre un conducteur non averti.

Chapitre 4

Détection d'intrusion pour les réseaux automobiles embarqués

Sommaire

Comme nous l'avons vu dans le chapitre 2, les réseaux automobiles actuels sont vulnérables aux attaques. En effet, un attaquant ayant obtenu un accès sur un réseau embarqué peut alors effectuer des actions de type lecture, interruption, modification ou fabrication de trames pour atteindre ses objectifs. Comme vu dans le chapitre 3, la sécurisation des systèmes embarqués est devenue une priorité pour les constructeurs ; nécessitant la mise en place de politiques de sécurité tenant compte des spécificités des systèmes embarqués dans les automobiles, tout en considérant les réseaux automobiles dans leur ensemble. Ainsi, en plus des techniques préventives visant à empêcher un attaquant de pénétrer le réseau, il apparaît également nécessaire de mettre en place des mécanismes surveillant le réseau « depuis l'intérieur », tels que des systèmes de détection d'intrusion. Or, comme nous l'avons vu dans le chapitre 1, les solutions de sécurité pour l'automobile doivent tenir compte des contraintes de compatibilité et d'autonomie, en plus des contraintes générales liées à la conception de systèmes embarqués.

Dans ce chapitre, nous présentons un système de détection d'intrusions pour les réseaux automobiles embarqués se basant sur le trafic réseau observé. Ce chapitre s'agence ainsi :

- La section 4.1 détaille la problématique, nous y décrivons les hypothèses que nous faisons au sujet des attaquants et présentons les objectifs de notre système.
- La section 4.2 décrit le fonctionnement général de notre système de détection d'intrusion afin de remplir les objectifs définis précédemment.
- Dans la section 4.3, nous formalisons notre méthode de génération de signatures d'attaque à partir de spécifications du système observé.
- Dans la section 4.4, nous évaluons la complexité (spatiale et temporelle) de notre système et confrontons en ces termes deux implémentations possibles de détection d'intrusion basées sur les signatures précédemment définies.
- Finalement, nous nous intéressons à la génération de séquences d'attaques minimales, qui peut s'avérer utile dans le cadre d'analyses *forensics*.

4.1 Problématique

Comme nous l'avons vu dans le chapitre 2, la littérature comprend désormais plusieurs exemples documentés d'attaques ciblant les réseaux automobiles embarqués (et plus spécifiquement CAN). De telles attaques vont d'une approche en « boîte noire » [HKD09], visant à découvrir le réseau embarqué et identifier les trames susceptibles d'être utilisées pour contrôler les ECU ciblés par de futures attaques [MV13], à des reprogrammations d'ECU via le réseau [KCR+10]. De plus, en 2011, Checkoway *et al.* [CMK+11] ont prouvé qu'il était possible d'obtenir un accès sur le réseau embarqué à distance en exploitant des vulnérabilités présentes dans les ECU responsables des communications avec l'extérieur. Ce faisant, ils furent alors capables de reproduire des attaques contre le réseau embarqué décrites dans [KCR+10], prenant ainsi le contrôle à distance sur les calculateurs embarqués.

4.1.1 Hypothèses d'attaque

Dans le cadre de nos travaux, nous supposons ainsi qu'un attaquant a réussi à obtenir un accès au réseau interne. Nous supposons alors que celui-ci va chercher à utiliser cet accès pour poursuivre son attaque contre le véhicule afin de perturber ou prendre le contrôle d'autres calculateurs accessibles via le réseau interne. Notre objectif est alors de détecter une attaque en cours en identifiant depuis le réseau des actions de l'attaquant observables ou des comportements inattendus d'un (sous)-système, découlant de ces actions. En d'autres termes, nos travaux visent à détecter une intrusion sur le réseau embarqué et non à identifier les origines de cette intrusion (attaque à distance, connexion sur le port OBD, sabotage en usine...). Si nous reprenons la classification présentée dans la section 2.4, nous considérons donc les attaques interagissant avec le réseau, et entraînant une compromission temporaire

ou durable de celui-ci. Les conditions relatives à l'origine de l'intrusion (portée et lien) ne rentrent pas en considération.

Nous supposons par ailleurs que l'attaquant peut avoir une connaissance avancée du réseau – telle que son architecture détaillée, ou la signification des données circulant sur les bus – sur lequel il s'est introduit. Nous considérons finalement que pour atteindre son objectif, l'attaquant va devoir interagir avec le réseau interne du véhicule afin de contrôler ou perturber des ECU distincts de son point d'entrée.

En fonction des moyens d'action de l'attaquant, ces attaques peuvent se traduire de différentes façon sur le réseau. En effet, si l'attaquant possède un contrôle total sur un ou des ECU émettant normalement les messages concernés par son attaque, il peut alors contrôler leurs transmissions et choisir précisément quelles trames seront transmises ou non, quel sera leur contenu et à quel moment la transmission aura lieu. En revanche, s'il contrôle un élément différent de ces ECU sur le réseau, il ne pourra pas contrôler aussi précisément la transmission de données sur le bus. Dans ce cas, les trames envoyées par l'attaquant cohabiteront alors sur le bus avec leurs versions légitimes, toujours émises par les ECU correspondants. L'attaquant ne peut pas non plus stopper ces transmissions légitimes, à moins de provoquer un déni de service sur le bus. Les deux cas précédents supposent que l'attaquant sait précisément ce qu'il doit faire pour mener son attaque à bien. Or, il est bien entendu également envisageable qu'un attaquant ne possède pas suffisamment de connaissances à propos du réseau sur lequel il s'est introduit et cherche à en apprendre plus, ou qu'il se trompe. Ses actions peuvent donc également être à l'origine de trafic incohérent.

Par conséquent, nous pouvons alors distinguer quatre types de comportements suspects sur le réseau :

1. Les trames ne respectant pas les spécifications du protocole réseau. Typiquement, dans le cas de CAN, cela concerne par exemple les trames comportant un identifiant inconnu, un champ de données de la mauvaise taille, un CRC incorrect, etc.

2. Des trames (ou séquences de trames) malveillantes émises périodiquement alors que le trafic légitime correspondant est toujours généré par le système. Par exemple, un attaquant émet périodiquement une trame commandant l'extinction des feux de croisement alors que le système émet des trames demandant leur allumage.

3. Des trames ou séquences malveillantes périodiques émises en remplacement des trames légitimes. Ces dernières ne sont alors plus émises sur le réseau.

4. Des trames ou séquences malveillantes émises ponctuellement, correspondant à des trames bien formées non périodiques dont l'émission est normalement conditionnée par un événement donné.

4.1.2 Objectifs

Nous décrivons ici les objectifs que l'on souhaite atteindre vis-à-vis des hypothèses d'attaque que nous venons d'établir, et en tenant compte des contraintes présentées dans le chapitre 1. Pour ce faire, nous nous concentrons sur quatre aspects en particulier : la source des données, la méthode de détection, les types d'attaques que l'on souhaite détecter et le comportement après détection.

Source des données

[LNJ08] décrit un IDS dont les règles se basent sur les spécifications du réseau (ils prennent pour référence le protocole CANopen [1]). Dans ce cas, chaque ECU est équipé d'un module chargé de vérifier si les trames émises ou consommées par cet ECU respectent ces règles. Certaines attaques peuvent nécessiter une coopération entre plusieurs modules pour être détectées. Un tel système présente l'avantage de pouvoir détecter les attaques de type lecture (cf. section 2.1.2.2), puisque les ECU eux-mêmes sont équipés de capteurs. Dans [MHT+12], chaque ECU doit détecter si ses trames sont également émises par d'autres éléments du réseau. Ainsi, il est possible de détecter (et même de contrer automatiquement via l'émission de trames d'erreur) des attaques de type fabrication sans nécessiter la mise en place de règles potentiellement complexes. En revanche, un ECU compromis n'est pas détecté tant qu'il n'émet que des trames qu'il est censé envoyer en temps normal. Par ailleurs, dans ces deux cas, la condition de devoir modifier les ECU fait que l'on perd en compatibilité (particulièrement vis-à-vis des COTS), ce qui entraîne de fait une augmentation des coûts lors d'un passage à l'échelle industrielle. Pour satisfaire au mieux les contraintes de compatibilité, nous ne souhaitons pas modifier les ECU déjà existants. Dans ce cas, la source des données est exclusivement le trafic réseau. En revanche, cela ne rend plus possible la détection d'actions de lecture illicites. Cependant, même si celles-ci sont considérées comme des attaques et peuvent peut-être parfois suffire à un attaquant pour atteindre ses objectifs, elles n'entraînent pas de modifications dans le fonctionnement du véhicule et n'ont donc pas d'impact sur la sécurité-innocuité de celui-ci. Par conséquent, nous faisons le choix de ne pas les traiter pour profiter du gain de compatibilité obtenu en contre partie.

Méthode de détection

Concernant la méthode de détection, la grande diversité des architectures automobiles (cf 1.2.1) couplée au relativement faible nombre d'attaques documentées contre les réseaux véhiculaires embarqués rend complexe une approche par scénarios. En effet, pour pouvoir être adaptable à un grand nombre de systèmes, une telle méthode doit être aisément généralisable. De plus, les mises à jour potentiellement requises par de telles approches ne sont pas (à l'heure actuelle) aussi aisées à faire dans une automobile que dans un ordinateur. En revanche, s'ils ont peu d'informations concernant des attaques avérées contre leurs systèmes embarqués, les

1. `http://www.can-cia.org/index.php?id=canopen`

constructeurs automobiles possèdent une connaissance étendue du fonctionnement de ces systèmes. Partir de cette connaissance pour définir un système de détection d'intrusion (et donc suivre une approche comportementale) représente une solution bien plus aisée à mettre en place. De plus, la tendance à l'unification des processus de conception logicielle (par exemple avec AUTOSAR) permet d'envisager la généralisation d'une telle procédure.

Couverture

Revenons sur les quatre types de comportements suspects décrits dans la section 4.1.1. Dans le cadre d'une approche comportementale, les deux premiers cas se révèlent simples à détecter grâce à un ensemble de règles. En effet, ces deux cas correspondent à un non-respect des spécifications du réseau :

- Une liste de spécifications peut permettre de détecter les trames erronées.
- De même, des trames malveillantes émises périodiquement en plus du trafic légitime peuvent être détectées grâce à l'augmentation de la fréquence d'apparition de celles-ci.

En revanche, dans les deux derniers cas, les messages envoyés par un attaquant peuvent être parfaitement conformes aux spécifications et respecter les contraintes temporelles. L'identification de telles trames parmi le trafic standard implique la mise en place de mécanismes permettant de confronter les informations fournies par une trame aux données précédemment observées. Nous désignons par la suite de tels mécanismes comme des mécanismes de corrélation.

Ces deux aspects (règles et corrélation) de la détection sont pris en compte dans [MGF10], via la description de 8 catégories de capteurs détectant différents types d'anomalies, dont certaines nécessitent une vérification de la cohérence du contenu des trames. Cependant, le fonctionnement des capteurs proprement dits n'est pas précisé.

Par ailleurs, [MA11] et [MV14] utilisent des approches basées sur un apprentissage du comportement du réseau. Dans ces cas, la corrélation se fait par rapport à ce qui a été observé lors de l'apprentissage, et non par rapport à ce qui a été vu depuis le début de l'observation. De plus, de telles méthodes sont plus propices aux faux-positifs vu qu'il n'est pas garanti que tous les cas d'utilisation possibles aient été pris en compte. À cet effet, [MA11] mentionne qu'une telle méthode pourrait être enrichie par la prise en compte de différents états du véhicule (à l'arrêt, démarrage, en train de rouler...) auxquels correspondraient différentes entropies. Cet aspect peut être considéré comme un mécanisme permettant d'établir une corrélation, mais aucune approche concrète n'a été proposée dans ce papier.

Comportement après détection

Nous nous plaçons dans les cas où un attaquant cherche à utiliser le réseau embarqué pour interagir avec des systèmes embarqués d'une voiture. Par conséquent, idéalement, un IDS surveillant le réseau doit pouvoir permettre d'intercepter une

attaque en cours afin que l'attaquant ne puisse pas atteindre ses objectifs. Cependant, comme nous considérons un système ne modifiant pas les ECU déjà présents sur le réseau, les moyens de réaction sont alors limités. Ainsi, en cas d'attaque détectée, il peut être possible de saturer le bus afin d'empêcher l'émission d'autres messages, comme envisagé dans [MV14]. Cela aura pour conséquence secondaire de faire passer de nombreux ECU du réseau dans un mode dégradé, ceux-ci ne pouvant plus obtenir d'informations via le bus. Cette contrepartie, certes toujours déplaisante pour le conducteur, peut dans certains cas s'avérer préférable à l'issue de l'attaque. La mise en place de mécanismes de réaction adaptés à l'intensité de l'attaque détectée devra faire l'objet de développements futurs (tout en s'assurant que de tels mécanismes ne peuvent à leur tour pas être utilisés par un attaquant). En l'état, nous nous concentrons ici sur la partie détection et considérons donc notre système comme passif vis-à-vis des attaques détectées.

4.2 Principe de fonctionnement

Nous allons maintenant nous attacher à décrire le fonctionnement du système de détection d'intrusion que nous proposons dans son ensemble, puis nous détaillerons une méthode permettant de déterminer la cohérence d'un message lors de sa transmission sur un bus.

4.2.1 Emplacement de l'IDS

Rappelons ici que le réseau embarqué d'une voiture consiste généralement en au moins deux sous-réseaux interconnectés via un ou plusieurs ECU jouant le rôle de passerelles. Afin de pouvoir récupérer le plus de données possible, notre IDS doit donc être connecté à tous les sous-réseaux nécessaires pour assurer la surveillance. Par conséquent, notre système peut être soit inclus dans une de ces passerelles, soit être un nouvel élément du réseau, connecté à tous les bus nécessaires. Il est à noter que dans ce dernier cas, cela crée un lien supplémentaire entre les sous-réseaux, ce qui peut représenter une augmentation des risques de sécurité. En effet, lors d'une attaque, les passerelles représentent des cibles de choix puisqu'elles permettent à un attaquant d'obtenir un accès sur de nouvelles portions du réseau. Une version distribuée de l'IDS, constituée de plusieurs modules placés dans les différents sous-réseaux est également envisageable. Cependant, cela implique la mise en place de canaux de communication entre ces différents modules, et donc la prise en compte des risques (si les canaux utilisent des bus existants, le système ne devient plus passif puisqu'il interagit avec le réseau surveillé) et/ou des coûts (en cas de mise en place d'un canal physique dédié) supplémentaires associés.

En résumé, si nous considérons une architecture réseau d'une automobile moderne, par exemple l'architecture de référence du projet EVITA (voir §3.1), les différents emplacements possibles pour un IDS représentés en figure 4.1 sont les suivants :

A. Sur un bus reliant plusieurs passerelles. Dans un tel cas, cela permet de surveiller les échanges entre les différents domaines (notamment entre les domaines multimédia/télécommunications, qui contiennent la majorité des points d'entrée pour une attaque, et le reste du véhicule). En revanche, cela ne donne pas accès aux messages transmis au sein des sous-réseaux placés derrière une passerelle.

B. Au sein d'une passerelle déjà existante. Ici, cela permet d'avoir accès aux messages circulant sur plusieurs bus. Cependant, une passerelle constituant une cible privilégiée lors d'une attaque (surtout si, comme dans notre figure, celle-ci sert également d'interface avec l'extérieur), une attention supplémentaire doit être apportée à la sécurisation de l'IDS au sein de cette passerelle afin qu'il ne soit pas corrompu lors d'une attaque (par exemple en assurant une isolation logicielle via des mécanismes de virtualisation).

C. Un ou plusieurs modules répartis dans tout le réseau. Ceux-ci peuvent opérer indépendamment ou communiquer entre eux.

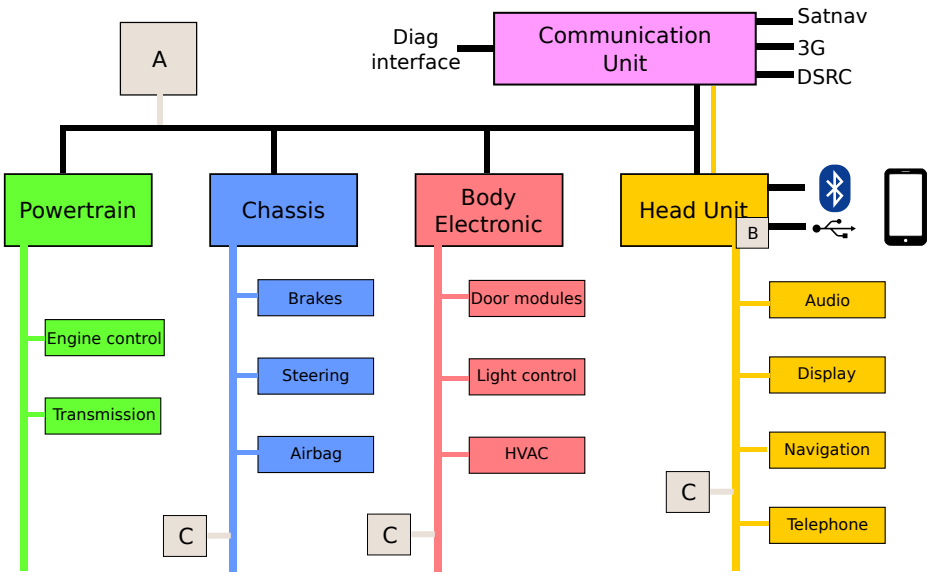


FIGURE 4.1 – Emplacements possibles d'un IDS dans une architecture automobile moderne type

4.2.2 Fonctionnement général

Au vu des problématiques évoquées précédemment, notre objectif est donc de proposer une méthode de détection d'intrusion depuis un point du réseau automobile embarqué, ne nécessitant donc pas ou peu de modifications du système existant,

pouvant être adaptée aisément à de nouvelles architectures et qui soit capable de prendre en compte le l'état du véhicule (ou d'un sous système) pour replacer un message dans son contexte. Comme vu en §4.1.2, certains comportements suspects peuvent être facilement détectés grâce à un système de règles issues des spécifications du protocole alors que d'autres nécessiteront d'effectuer une corrélation entre les données observées pour pouvoir être détectés. Comme nous souhaitons pouvoir détecter des attaques appartenant à ces deux catégories, nous devrons effectuer deux analyses sur les messages. Le processus de détection complet est ainsi résumé en figure 4.2. Tout d'abord, lorsqu'une nouvelle trame est observée sur un bus, une première étape consiste à identifier le type de message qu'elle contient (dans le cas de CAN, cela se fait via l'identifiant) pour éventuellement en extraire les signaux nécessaires. Ensuite, la première partie de la détection d'intrusion est effectuée en vérifiant que la trame respecte bien les règles préalablement établies par le protocole réseau et/ou les spécifications du constructeur. Lorsqu'une règle n'est pas respectée, cette violation est enregistrée et une alerte peut éventuellement être levée en fonction de la fréquence et de la sévérité des violations précédemment observées. Par suite, les analyses relatives au contexte d'émission de la trame prennent place, avant de revenir à l'étape 1 pour la trame suivante. Les mécanismes permettant ces analyses, que nous détaillons dans les sections 4.3 et 4.4, reposent sur des automates finis.

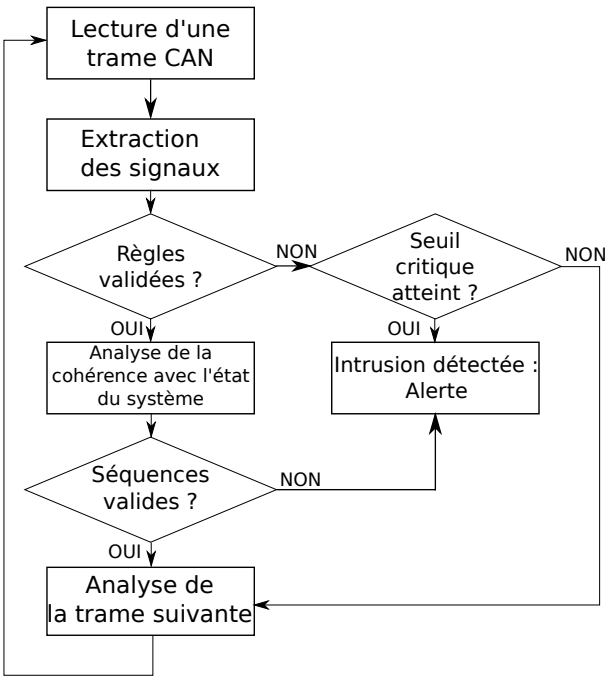


FIGURE 4.2 – Schéma du processus de détection d'intrusion

Règles de contrôle

Comme nous l'avons vu précédemment, les attaques ne respectant pas les spécifications du réseau peuvent être détectées via un ensemble de règles de contrôle. Un exemple de génération de telles règles à partir de spécifications est donné dans [LNJ08]. Dans un premier temps, nous nous restreignons aux cas suivants :

- Identifiant inconnu : l'identifiant d'une trame ne fait pas partie d'une liste déterminée.
- Bus d'émission incorrect : une trame est émise sur un bus où elle n'est pas censée se trouver.
- Taille du champ de données incorrecte : le champ de données ne correspond pas à ce qui est spécifié.
- Fréquence incorrecte : dans le cadre d'une trame périodique, le temps qui s'est écoulé entre deux observations de cette trame doit respecter les spécifications.

Définition d'un contexte d'émission

Pour détecter des attaques relevant des deux dernières catégories, de telles règles de contrôle s'avèrent en revanche insuffisantes. En effet, les trames émises dans ces types d'attaques seront potentiellement conformes à toutes les règles évoquées précédemment. Pour de tels cas, il est nécessaire de se baser sur le contexte d'émission des trames. En d'autres termes, nous avons besoin de récupérer et de recouper des informations provenant de messages observés précédemment afin de déterminer l'état actuel de la voiture (ou du moins du sous-système considéré). Le système de détection utilise alors ces données pour décider si la trame ou le message qu'elle contient sont cohérents avec le contexte ainsi défini.

Pour définir un tel contexte, et en fonction des cas, la corrélation peut se faire vis-à-vis de deux paramètres :

- Redondance : des données similaires qui proviennent de sources distinctes. Par exemple, un message contenant la vitesse du véhicule peut provenir du système de freinage mais également du système de navigation par satellite. De même, des mécanismes de redondance peuvent être présents pour des raisons de *safety* : une même information peut ainsi être transmise sur plusieurs trames distinctes.
- Cohérence : l'historique des trames précédemment émises ou consommées par le sous-système considéré permet de déterminer quelle(s) trame(s) est (sont) attendue(s) par la suite.

Dans les systèmes d'informations et réseaux traditionnels, les IDS doivent tenir compte du fait qu'ils ne possèdent pas nécessairement une connaissance détaillée du système qu'ils surveillent. Ainsi, ils peuvent ne pas connaître sa structure précise à tout instant (des machines peuvent être connectées ou retirées du réseau) ni l'étendue des comportements possibles (par exemple, les interactions homme-machine possibles peuvent très variées). Ce n'est pas le cas concernant le réseau interne d'une voiture : l'architecture est a priori figée et ne changera pas durant la vie du véhicule, les interactions attendues entre les ECU sont prévisibles et les

actions humaines sont limitées. Ce plus haut niveau de déterminisme du réseau peut ainsi être utilisé à notre avantage. Notre objectif devient alors de détecter les trames émises qui ne sont pas conformes au comportement prévu.

Déclenchements d'alertes

Une apparition ponctuelle de telles anomalies ne témoigne pas nécessairement d'une attaque. En effet, des perturbations électromagnétiques peuvent altérer le contenu d'une trame, un capteur peut se tromper ponctuellement ou l'émission d'une trame peut être retardée. Il devient alors plus pertinent de s'intéresser à la fréquence d'apparition de telles anomalies avant de lever une alerte. Pour affiner la détection, nous envisageons une approche similaire à celle proposée par Müter *et al.*[MGF10], reposant sur une fenêtre glissante et 3 seuils correspondant à des niveaux de criticité croissants (*important, critical, severe*). La condition représentant le dépassement d'un seuil à un instant t est la suivante :

$$\frac{n}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} X_{it} w_i > T'$$

où :

- $X_{it} = \sum_{j=t-SLW}^{t} S_{ij}$ décrit la somme des incidents identifiés par un capteur S_i durant une fenêtre de temps de longueur SLW.
- w_i représente la pondération attribuée au capteur S_i.
- T' est la valeur (normalisée) du seuil correspondant à un des 3 niveaux de criticité.

La figure 4.3 illustre ce principe. Dans ce cas, un seuil est fixé à 4 anomalies (toutes pondérées de la même manière) minimum détectées durant une fenêtre de taille SLW. Avant $t2$, aucun alerte n'est ainsi levée malgré la détection d'une anomalie à $t1$ car la fenêtre glissante ne contient jamais 4 alertes. En revanche, à $t2$, ce seuil est atteint et une alerte est levée.

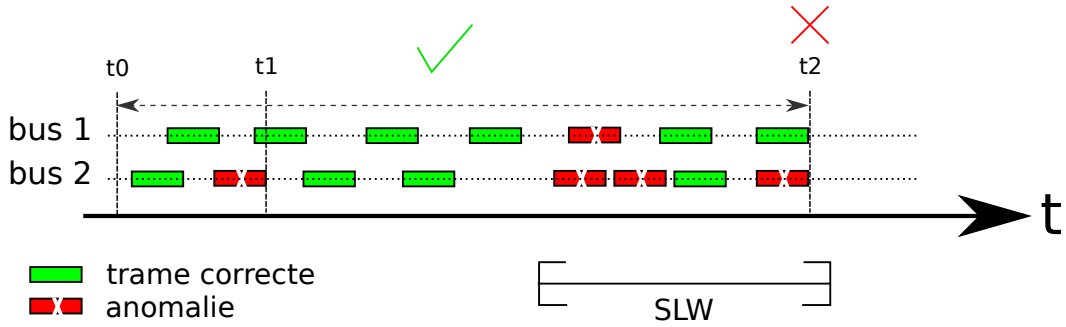


FIGURE 4.3 – Principe de la fenêtre glissante décrite par [MGF10]

La taille de SLW, les valeurs des seuils et les pondérations w_i ne sont pas fixées. Elles dépendront des exigences du constructeur (en termes de sécurité immunité et

innocuité) vis-à-vis des systèmes concernés.

4.2.3 Synthèse

En résumé, si nous reprenons la classification des systèmes de détection d'intrusions présentée en §3.4.1, les caractéristiques de notre système sont les suivantes :

- **Source des données** : Comme détaillé en 4.2.1, notre IDS récupère ses données sur le réseau.
- **Méthode de détection** : Comme justifié en 4.1.2, l'approche utilisée pour la détection est comportementale.
- **Analyse des données** : Dans un premier temps, nous ne considérons pas de cas où l'IDS est distribué sur le réseau, l'analyse des données se fait donc localement.
- **Fréquence de l'analyse** : L'objectif est de pouvoir effectuer une analyse continue. En effet, comme nous nous intéressons particulièrement aux attaques visant à modifier le comportement des ECU à travers le réseau, il peut être crucial de réagir au plus vite afin de préserver l'intégrité du véhicule et de ses passagers si le système ciblé par l'attaque est critique.
- **Comportement après détection** : Dans un premier temps, nous nous contentons d'un système passif levant une alerte en cas d'intrusion détectée. La mise en place de mécanismes visant à contrer une attaque en cours est envisagée à plus long terme.

4.3 Formalisation d'une méthode de corrélation

Nous décrivons maintenant l'approche que nous proposons afin de formaliser un mécanisme de corrélation de messages requis pour implémenter la deuxième étape de la détection d'intrusion décrite en 4.2.2. Dans le cadre de notre approche, nous nous plaçons du point de vue d'un constructeur automobile. Celui-ci possède une connaissance avancée des systèmes embarqués dans ses véhicules et peut utiliser les spécifications des ECU afin de modéliser le comportement des systèmes qu'il souhaite surveiller. Cependant, utiliser directement de tels modèles générés à partir des spécifications pour analyser les messages circulant sur les bus requiert un bon niveau de synchronisation entre l'IDS et les autres éléments du réseau. En effet, nous souhaitons pouvoir poursuivre la surveillance du réseau après une première détection (par exemple, si la séquence détectée ne menace pas un système critique, nous pouvons considérer qu'il n'est pas nécessaire d'alerter immédiatement le conducteur ni de demander un arrêt du véhicule). Or, si notre système a identifié une déviation par rapport au comportement attendu, il ne connaît alors plus avec certitude l'état dans lequel se trouvent les systèmes surveillés. Une solution à ce problème pourrait être de rendre notre IDS capable d'interroger les autres ECU afin de connaître leur état actuel. Cependant, ceci rendrait beaucoup plus complexe l'implémentation de l'IDS au sein du véhicule et irait à l'encontre de notre objectif visant à ne pas modifier en profondeur l'architecture existante.

Par conséquent, nous avons choisi d'employer une méthode de détection ne nécessitant pas une forte synchronisation avec le système observé. Celle-ci repose sur la définition d'un langage décrivant des séquences de messages interdites, qui correspondent aux comportements non désirés du système surveillé. Nous utilisons alors des automates pour détecter ces séquences lors de l'observation du système.

Dans le reste de cette section, un **système** désignera l'ensemble des ECU impliqués dans la réalisation d'une tâche donnée. Par exemple, le système « freinage » (fictif) comprend les ECU contrôlant les pédales, le frein à main, les freins eux-mêmes et les feux de freinage. Un ECU peut être impliqué dans plusieurs tâches, et donc faire partie de plusieurs systèmes.

4.3.1 Automates finis

Les ECU sont souvent spécifiés (entre autres) via des machines à états. Ainsi, les trames qu'ils peuvent émettre sur le réseau à un moment donné dépendent de leur état actuel, et ce dernier est également conditionné par les trames que l'ECU aura consommées. Pour chaque ECU d'un système, nous pouvons dès lors définir un langage L_{ECU} comme suit :

- Les symboles sont les différents signaux (concernant le système dont il est question) qu'il transmet ou reçoit via le réseau. Un alphabet, nommé Σ_{ECU} est l'ensemble de tous ces symboles.
- Les mots sont alors toutes les séquences valides de symboles, c'est-à-dire les séquences pouvant être observées lors d'un usage légitime du système. Un tel langage est préfixe-clos, c'est à dire que tout préfixe d'un mot valide du langage est également valide.

À partir de ce langage, nous pouvons créer un **automate fini** le reconnaissant. Cet automate constitue ainsi un modèle du comportement de l'ECU tel qu'il est perçu depuis le réseau. Nous rappelons qu'un automate fini consiste en un quintuplet $(Q, \Sigma, \delta, I, F)$, où :

- Q est l'ensemble fini des états.
- Σ est l'alphabet fini.
- $\delta : Q \times \Sigma \to P(Q)$, où $P(Q)$ est un sous-ensemble de Q, est la fonction de transition
- I est l'ensemble des états initiaux.
- $F \subseteq Q$ est l'ensemble des états finals (ou terminaux).

De plus, un **automate fini déterministe** (AFD) est un automate fini qui vérifie les deux conditions suivantes :

- Il possède un état initial unique : $I = \{q_0\}$
- La fonction de transition est déterministe, c'est-à-dire que pour un état q et un symbole s donnés, il existe au plus une seule transition étiquetée par s.

Un automate est dit **complet** si chacun de ses états possède (au moins) une transition sortante pour chaque symbole de l'alphabet. Un état q est dit **accessible** s'il existe un chemin allant d'un état initial à q. Un état q est dit **coaccessible** s'il existe un chemin allant de q à un état final. Lorsqu'un automate contient des

états non-coaccessibles, ceux-ci peuvent-être rassemblés en un unique état, que nous qualifierons d'état **puits**.

Dans le reste de ce chapitre, tout automate est considéré **complet** [2].

L'automate fini déterministe du système, noté A_{sys} est défini comme la composition parallèle synchronisée de tous les automates des ECU compris dans le système. Le langage du système décrit par A_{sys} est noté L_{sys} et leur alphabet Σ_{sys}.

La composition parallèle synchronisée de deux AFD complets A et B est définie ainsi :

$$A \parallel B = (Q_A \times Q_B, \Sigma_A \cup \Sigma_B, \delta, I_A \times I_B, F_A \times F_B)$$

où

$$\delta(q_A.q_B, s) = \begin{cases} \delta(q_A, s).\delta(q_B, s) \text{ si } s \in \Sigma_A \cap \Sigma_B \\ \delta(q_A, s).q_B \text{ si } s \in \Sigma_A \wedge s \notin \Sigma_B \\ q_A.\delta(q_B, s) \text{ si } s \notin \Sigma_A \wedge s \in \Sigma_B \\ q_A.q_B \text{ sinon} \end{cases}$$

Par construction, L_{sys} est préfixe-clos car il est la composition de langages préfixe-clos. Dans A_{sys}, tout état différent de l'état puits est donc un état final.

4.3.2 Génération d'un langage d'attaques

Nous rappelons que notre objectif est de détecter une attaque en cours sur le réseau en vérifiant si une trame transmise sur un bus est cohérente vis-à-vis du comportement précédemment observé du système. En d'autres termes, nous voulons détecter les cas où une trame génère un symbole faisant partie d'une séquence qui ne constitue pas une séquence valide de L_{sys}. Une séquence invalide (ou interdite) peut être définie comme une séquence qui est valide jusqu'à un certain point où un symbole inattendu invalide celle-ci. Formellement, si w est un mot invalide de L_{sys} et $w = s_0 s_1..s_n$ où tous les s_i représentent des symboles de Σ_{sys}, alors :

$$\exists v \in L_{sys}, \exists k \in [0, n-1], s_0..s_k \in pref(v) \wedge s_0..s_{k+1} \notin L_{sys}$$

où $pref(v)$ est l'ensemble de tous les préfixes de v.

Nous décrivons ci-après les différentes étapes de la méthode que nous avons employée pour générer l'ensemble des séquences invalides.

1. Nous construisons l'ensemble de tous les mots qui commencent comme des mots de L_{sys} (or, L_{sys} étant préfixe-clos, cet ensemble est égal à L_{sys}) concaténés avec un symbole supplémentaire de Σ_{sys}.
2. Appelons alors $\overline{L_{sys}}$ le complémentaire L_{sys}. $\overline{L_{sys}}$ contient ainsi tous les mots sur Σ_{sys} n'étant pas dans L_{sys} et par conséquent toutes les séquences invalides pour le système concerné.

2. Il est aisé de rendre complet un automate fini en créant un état supplémentaire, non final, vers lequel pointeront toutes les transitions manquantes de chaque état de l'automate original.

3. En calculant l'intersection des ensembles décrits aux étapes 1 et 2, nous obtenons alors l'ensemble suivant :

$$\{w \in \overline{L_{sys}} | \exists (x, s) \in L_{sys} \times \Sigma_{sys}, w = x.s\}$$

Cet ensemble correspond à $(L_{sys}.\Sigma_{sys}) \cap \overline{L_{sys}}$ et contient toutes les séquences invalides tronquées après la première déviation par rapport à un comportement normal. Nous avons alors créé un langage pouvant théoriquement être utilisé pour détecter toute déviation par rapport au fonctionnement normal du système, tant qu'il est en mesure de connaître tout l'historique des messages émis et reçus depuis son initialisation.

4. Cependant, nous voulons pouvoir reprendre l'observation suite à une détection (sans avoir à réinitialiser le système, ni à redémarrer la voiture). Par conséquent, nous souhaitons pouvoir détecter une attaque dès que possible sans pour autant nécessairement posséder l'intégralité de la séquence décrite par le système depuis son initialisation. En d'autres termes, nous voulons pouvoir détecter une attaque en ne possédant qu'un suffixe de la séquence totale. Formellement, cela revient à vouloir identifier des séquences interdites appartenant à l'ensemble $Suf(L_{sys}.\Sigma_{sys} \cap \overline{L_{sys}})$, où $Suf(L)$ est l'ensemble de tous les suffixes des mots du langage L.

5. En revanche, cet ensemble contient également des mots qui constituent des portions de séquences valides de L_{sys} (c'est-à-dire que ces mots appartiennent à l'ensemble des radicaux (*factors*) de L_{sys}, noté $Fact(L_{sys})$). Lever une alerte lorsque ces mots sont identifiés générerait alors un faux-positif. Pour éviter cela, nous devons donc retirer de cet ensemble tous les mots appartenant à $Fact(L_{sys})$. Or, notons que $Fact(L_{sys}) = Suf(Pref(L_{sys})) = Suf(L_{sys})$ car L_{sys} est préfixe-clos. Nous obtenons finalement l'ensemble suivant :

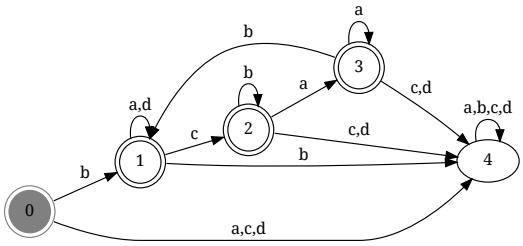
$$Suf(L_{sys}.\Sigma_{sys} \cap \overline{L_{sys}}) \cap \overline{Suf(L_{sys})}$$

Ce dernier ensemble, que nous appelons $S_{attacks}$, correspond au langage des anomalies possibles observables sur le réseau pour un système donné. De plus, $S_{attacks}$ est créé à partir du langage rationnel L_{sys} et les langages rationnels sont fermés pour les opérations utilisées lors de cette génération. Par conséquent, $S_{attacks}$ est également un langage rationnel, noté $L_{attacks}$.

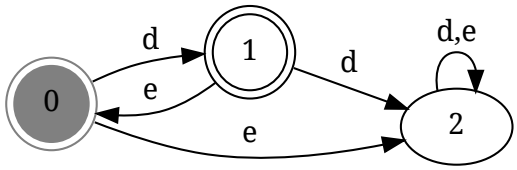
Exemple

Dans les figures suivantes, les états grisés sont des états initiaux et les états possédant une double bordure sont des états finals. Considérons les automates de la figure 4.4, qui représentent 3 ECU fictifs, constituant un système. Les transitions correspondent à des signaux transmis sur le réseau (comme il n'y a pas de notion de source ou de destination dans une trame CAN, ces deux cas sont identiques du point de vue d'un observateur placé sur le réseau). L'automate minimal résultant

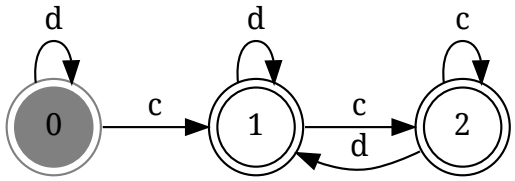
de leur composition parallèle synchronisée est donné en figure 4.5. Cet automate reconnaît le langage L_{sys}. Nous présentons en figure 4.6 l'automate déterministe $A_{attacks}$ qui reconnaît $L_{attacks}$. Dans cette figure, nous avons mis en évidence la séquence interdite *acd*.



(a)



(b)



(c)

FIGURE 4.4 – Automates des ECU initiaux

4.4 Évaluation de la complexité

Idéalement, qui plus est dans un contexte comme celui d'une voiture, le traitement d'une trame par l'IDS doit se faire le plus rapidement possible. En effet, une détection prenant trop de temps pourrait avoir des conséquences graves si l'attaque alors en cours aboutit avant d'avoir été détectée. Par ailleurs, les contraintes liées au développement de systèmes embarqués (voir 1.1.1) font qu'il est important de limiter les besoins en mémoire d'une application. Ces deux contraintes peuvent être exprimées en termes de complexité, temporelle et spatiale.

4.4.1 Méthode « naïve » basée sur les AFD

Les automates déterministes offrent des performances rapides pour détecter une signature. En effet, le parcours d'un AFD se fait en temps constant, puisqu'il n'existe qu'une progression possible pour chaque couple (état, transition). Une implémentation « naïve » de la méthode décrite précédemment consisterait alors à créer un

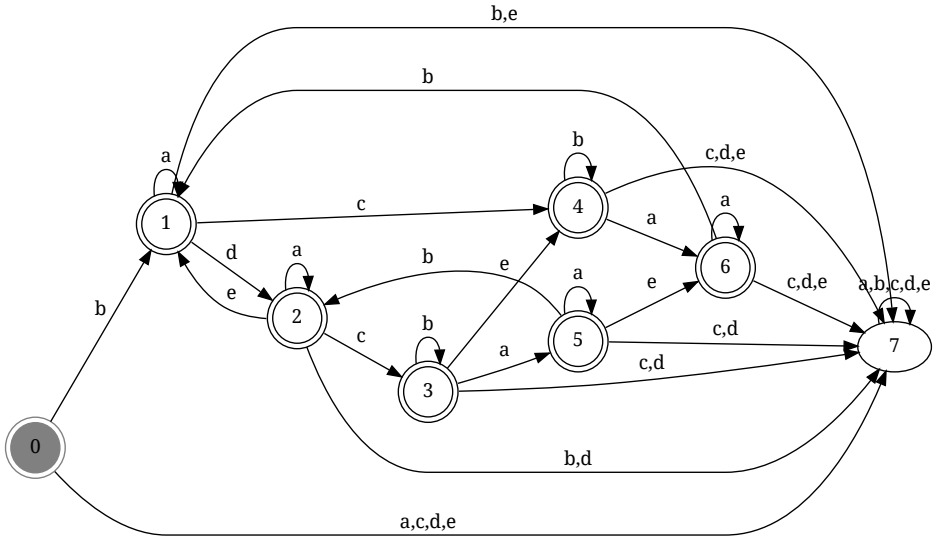


FIGURE 4.5 – Composition parallèle des automates (a), (b) et (c)

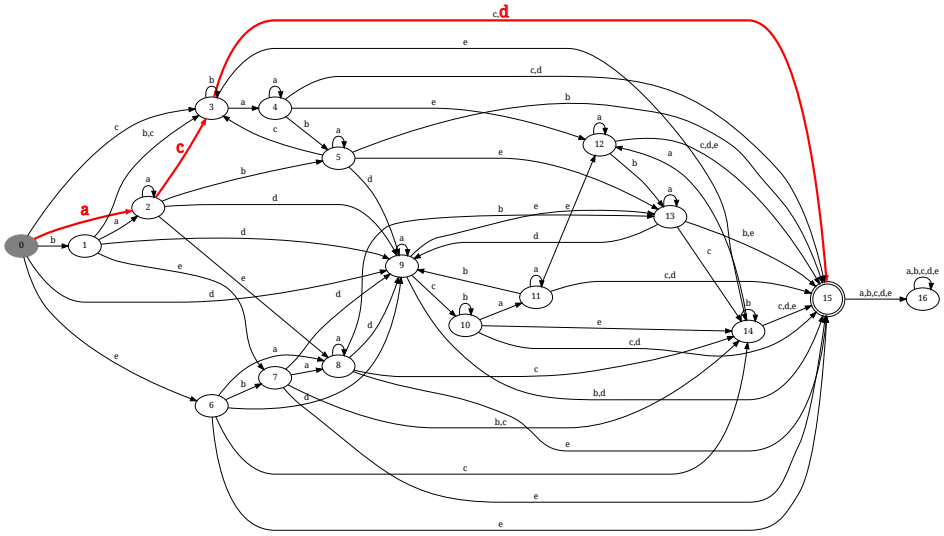


FIGURE 4.6 – Automate $A_{attacks}$

AFD $A_{attacks}$ correspondant au langage $L_{attacks}$ pour effectuer la détection. Une façon simple de représenter un AFD est d'utiliser un tableau à deux dimensions où chaque ligne correspond à un état et chaque colonne à un symbole étiquetant une transition. L'algorithme de détection se résume alors comme suit :

1. Au début de l'observation, l'IDS se trouve dans l'état initial de l'automate.
2. À chaque fois qu'un nouveau symbole est lu, le système parcourt $A_{attacks}$ en conséquence en prenant la transition correspondante.
3. Quand l'état final est atteint, une alerte est levée. L'observation peut alors reprendre dans l'état initial de $A_{attacks}$.
4. Éventuellement, si un problème intervient durant l'observation (par exemple, aucune trame n'est observée pendant une durée trop longue), l'IDS considère que son historique n'est plus valide et recommence une observation dans l'état initial.

Dans notre cas, nous estimons la complexité spatiale d'un algorithme en calculant la taille (c'est-à-dire le nombre d'états) maximale d'un automate généré par celui-ci. Les complexités dans le pire cas pour les opérations utilisées dans la méthode décrite en 4.3.2, sont ainsi résumées dans la table 4.1 (des preuves plus détaillées peuvent par exemple être trouvées dans [Yu01] et [BJZ10]). Dès lors, si nous supposons que le langage L_{sys} peut-être décrit par un AFD A_{sys} possédant n états, alors :

1. L_{sys} étant par définition clos par les préfixes, la complexité spatiale de $L_{sys}.\Sigma_{sys}$ est de $n + 2$ dans le pire cas. En effet, un AFD reconnaissant $L_{sys}.\Sigma_{sys}$ peut être construit à partir d'un automate reconnaissant L_{sys} auquel on ajoutera au plus deux états. Tout d'abord, nous créons un nouvel état initial dont les transitions sortantes sont identiques à celles de l'état initial original mais ne possédant pas de transitions entrantes (l'état initial original perd alors son statut). Ensuite, nous retirons toutes les transitions réflexives de l'état puits, que nous ajoutons à l'ensemble des états finals. Un nouvel état puits, atteignable uniquement depuis l'ancien état puits est alors ajouté pour compléter l'automate.
2. La complexité spatiale de $\overline{L_{sys}}$ est n (cf. table 4.1).
3. Par conséquent, dans le pire des cas, la complexité spatiale de $(L_{sys}.\Sigma_{sys}) \cap \overline{L_{sys}}$ devrait être de l'ordre de $O(n^2)$. Cependant, comme les deux côtés de l'intersection dérivent du même langage L_{sys}, ils possèdent des points communs. En effet, en appliquant une transformation similaire à celle décrite à l'étape 2 à l'automate décrivant $\overline{L_{sys}}$, mais sans modifier l'ensemble des états finals, nous obtenons un automate qui accepte toujours $\overline{L_{sys}}$ mais qui a la même structure que celui décrit en 2. Dès lors, l'intersection de ces deux automates similaires peut être construite en créant un automate possédant la même structure que ceux-ci mais pour lequel l'ensemble des états finals correspond à l'intersection des deux ensembles correspondants. Sa complexité spatiale est ainsi de $n + 2$.

4. Une façon simple de construire l'automate représentant l'ensemble des suffixes d'un langage régulier est de prendre un AFD représentant ce langage dans lequel tout état coaccessible devient un état initial. Cependant, l'automate résultant de ce procédé, possédant plusieurs états initiaux, n'est plus déterministe. La déterminisation de cet automate est possible, mais dans le pire cas, l'AFD alors obtenu peut posséder 2^n états.

Par conséquent, un AFD $A_{attacks}$ ainsi construit possédera potentiellement un nombre d'états exponentiellement supérieur à celui de l'automate A_{sys} initial. Ceci devient rapidement problématique en termes de stockage, ne serait-ce qu'avec un système possédant une centaine d'états.

Automate	Taille (pire cas)
L_1	m
L_2	n
$\overline{L_1}$	m
$L_1 \cap L_2$	mn
$L_1 \cup L_2$	mn
$Pref(L_1)$	m
$Suf(L_1)$	m (non déterministe)
$DFA(L_1)$ si L_1 non déterministe	2^m

TABLE 4.1 – Complexité spatiale pour des opérations basiques sur des automates finis

4.4.2 Réduction de la complexité spatiale

Afin de pouvoir traiter les cas pour lesquels la complexité spatiale est trop importante, nous proposons la méthode suivante, qui ne nécessite pas de calculer l'automate $A_{attacks}$.

Soit

$$L_{left} = Suf(L_{sys}.\Sigma_{sys} \cap \overline{L_{sys}})$$

et

$$L_{right} = \overline{Suf(L_{sys})}$$

Nous avons alors $L_{attacks} = L_{left} \cap L_{right}$.

Par définition, L_{right} contient tous les mots de Σ^* qui ne sont pas des radicaux de L_{sys}. L'intersection de L_{right} avec L_{left} permet de décrire le fait que la détection s'arrête (et/ou redémarre depuis l'état initial) lorsque la première déviation confirmée par rapport à tout comportement potentiellement légitime a été observée. Par conséquent, nous pouvons détecter les mots de $L_{attacks}$ en considérant simplement un automate qui reconnaît L_{right} et en arrêtant l'observation lorsque l'état final est atteint la première fois. Cependant, comme $L_{right} = \overline{Suf(L_{sys})}$, utiliser un AFD

pour effectuer la détection poserait également les problèmes de passage à l'échelle décrits précédemment.

Soit A_{right} un AFD qui reconnaît le langage L_{sys}. A_{right} a ainsi une taille en $O(n)$. Considérons l'ensemble S_{right} qui contient initialement tous les états de A_{right}. Cet ensemble représente en fait tous les états possibles dans lesquels le système observé peut se trouver depuis que son observation a commencé. À chaque fois qu'un nouveau symbole vient s'ajouter à la séquence observée, un nouvel ensemble est créé. Celui-ci contient les états pouvant être atteints depuis les états contenus dans S_{right} en prenant les transitions étiquetées par ce symbole. S_{right} devient alors ce nouvel ensemble pour la prochaine itération. Ainsi, avec cette méthode, nous pouvons parcourir A_{right} en considérant tous ses états comme des états initiaux.

L'algorithme 1 présente la procédure utilisée pour détecter une intrusion en utilisant l'ensemble S_{right} décrit précédemment.

Dans cet algorithme, la procédure **delta**(A, s, x) renvoie l'état atteint lorsque l'on prend la transition étiquetée par le symbole x depuis l'état s dans l'automate A et la procédure **isFinal**(s) renvoie $True$ si l'état s est final. Le principe de cet algorithme est le suivant :

1. À chaque fois qu'une trame contient des informations correspondant à un symbole de Σ_{sys}, le contenu de S_{right} est mis à jour en fonction de la fonction de transition de A_{right}.

2. S_{right} représente les parcours possibles dans $A_{right} = L_{sys}$ depuis le début de l'observation. Cependant, nous sommes en fait intéressés par les éléments de l'ensemble $\overline{Suf(L_{sys})}$. Par conséquent, la séquence observée est dans $\overline{Suf(L_{sys})}$ si et seulement si elle ne constitue pas un suffixe de L_{sys}, c'est-à-dire si et seulement si aucun élément de S_{right} ne correspond à un état final de A_{right}.

3. Dès que cette condition est vérifiée (donc si tous les états de S_{right} ne sont pas finals), alors la séquence que nous avons observée est un mot de L_{right}, et donc dans L_{attack}. Dans le cas contraire, l'observation continue.

Ce faisant, nous sommes alors capables de déterminer si une séquence observée constitue un mot de $L_{attacks}$ sans avoir construit l'AFD $A_{attacks}$ correspondant. Nous estimons la complexité spatiale de cette méthode de la façon suivante :

- $n \times |\Sigma_{sys}|$ pour A_{right}
- n pour S_{right}

Ce qui fait un total de

$$n(1 + |\Sigma_{sys}|)$$

La contrepartie de cette méthode est une augmentation de la complexité temporelle : pour chaque itération, nous exécutons la fonction delta au plus n fois pour A_{right} (une fois pour chaque élément de S_{right}) au lieu d'une unique fois si nous utilisions un automate $A_{attacks}$ déterministe.

Algorithm 1 Détection d'intrusion en ligne

```
1: procedure DETECT(S_right, x)
2:     newS_right ← []
3:     counter ← 0
4:     for i ← 0, size(S_right) − 1 do
5:         nextState ← delta(A_right, S_right[i]
6:         if nextState ∉ NewS_right then
7:             newS_right.append(nextState)
8:         end if
9:         if ¬isFinal(newS_right[i]) then
10:            counter ← counter + 1
11:        end if
12:    end for
13:    detection ← (counter = size(NewS_right))
14:    S_right ← newS_right
15:    return detection
16: end procedure
```

Exemple

L'implémentation de l'algorithme1 est illustrée par les figures 4.7 et 4.8 : La figure 4.7 présente l'automate A_{right} correspondants au A_{sys} de la figure 4.5. Dans ce cas, A_{sys} étant clos par les préfixes, on constate que $A_{right} = A_{sys}$. La figure 4.8 présente une exécution de l'algorithme 1 utilisant l'automate A_{right} de la figure 4.7 :

1. Tout d'abord, S_{right} est initialisé avec l'ensemble des états de A_{right}.
2. Le symbole 'a' est lu : pour chaque état dans S_{right}, nous récupérons alors le nouvel état atteint lorsque la transition sortante de celui-ci, étiquetée par 'a', est prise dans l'automate A_{right}. Remarquons que celui-ci est ici équivalent au A_{sys} de la figure 4.5 car A_{sys} est clos par les préfixes. Un nouvel ensemble est alors créé, qui deviendra alors S_{right} pour l'étape suivante.
3. Tant que la condition décrite dans l'algorithme 1 n'est pas satisfaite, l'observation continue et le symbole suivant est lu (ici, 'c').
4. En revanche, quand le symbole 'd' est lu, nous sommes alors dans une situation où tous les états de S_{right} ne sont pas finals (les cases rouges). Dans une telle situation, cela signifie que la séquence observée jusqu'à présent ('acd') est une séquence interdite. Une alerte est alors levée. Si l'on souhaite poursuivre l'analyse, il faudra alors réinitialiser S_{right} avec l'ensemble des états de A_{right}.

4.4.3 Passage à l'échelle

Comme mentionné précédemment, utiliser un AFD pour représenter $L_{attacks}$ peut s'avérer problématique puisque sa taille peut augmenter de manière exponentielle avec celle de A_{sys}. Nous avons proposé une solution alternative basée sur

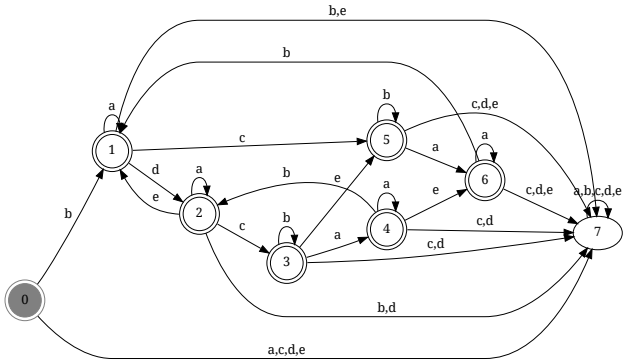


FIGURE 4.7 – A_{right}

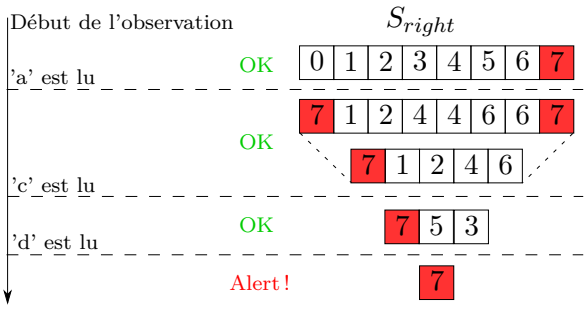


FIGURE 4.8 – Une exécution de l'algorithme 1 dans le cas de la séquence *acd*

l'emploi de l'automate déterministe A_{right}, plus petit, et de l'ensemble S_{right} qui lui est associé. Cependant, cette solution, si elle occupe moins de mémoire dans le pire cas, possède une complexité temporelle supérieure. Afin d'estimer l'impact que cela représente dans le cas de systèmes complexes (c'est-à-dire avec un grand nombre d'états), nous avons arbitrairement fixé une taille d'alphabet (20 symboles) et généré aléatoirement des AFD de différentes tailles utilisant cet alphabet. Chaque automate ainsi généré est alors utilisé comme un A_{sys} à partir duquel nous dérivons le A_{right} correspondant. Pour chaque échantillon, nous générons aléatoirement une séquence prévue pour déclencher une alerte lorsque le dixième symbole est atteint et analysons alors cette séquence via la méthode décrite en 4.4.2. Au début de chaque itération (donc à chaque fois qu'un nouveau symbole va être analysé), nous avons compté le nombre d'éléments compris dans S_{right}. Les valeurs obtenues pour des automates A_{sys} constitués de 10, 100 et 1000 états (respectivement les courbes rouges, vertes et bleues) sont illustrés en figure 4.9. Nous constatons ainsi que la taille de S_{right} diminue exponentiellement à chaque étape. Par conséquent, il apparaît que cette méthode peut s'avérer trop longue pour détecter une anomalie se produisant très rapidement après le début d'une observation. En revanche, après un petit nombre d'itérations, la taille de S_{right} aura fortement diminué, et les analyses seront alors bien plus performantes.

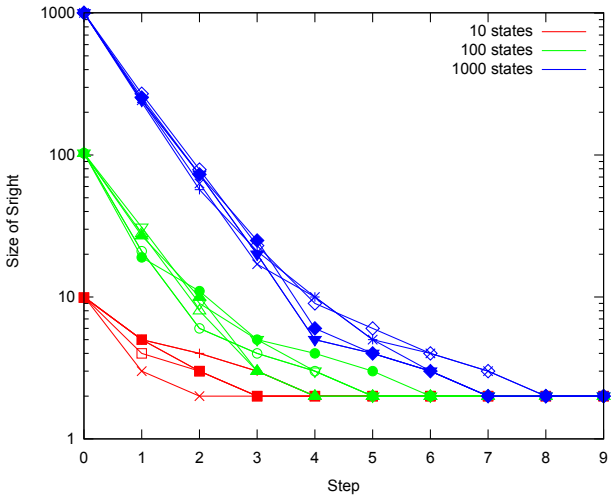


FIGURE 4.9 – Évolution de la taille de S_{right} durant une observation pour différentes tailles de A_{sys}

4.5 Énumération des séquences d'attaque minimales

Les mots du langage $L_{attacks}$ sont caractérisés par le fait qu'ils représentent tous des séquences dont le dernier symbole entraîne une déviation par rapport à toute séquence de $Suf(L_{sys})$. Cependant, ces mots ne constituent pas nécessairement des séquences minimales. Par exemple, si *abc* est un mot de $L_{attacks}$ et que *ababa* est un mot de $Suf(L_{sys})$, alors *abababc* sera également un mot de $L_{attacks}$. Lors d'une détection en ligne, ce fait n'a pas d'importance. En effet, c'est bien la première déviation par rapport à un comportement normal qui nous intéresse et il peut être intéressant de garder une trace de la séquence entière afin de voir a posteriori dans quelle situation l'attaque s'est déroulée. Définir l'ensemble des séquences d'attaque minimales peut en revanche avoir de l'intérêt dans le cadre d'analyses *forensics*. Par exemple, suite à un accident ou un dysfonctionnement d'un système donné dont on ne connaît pas la cause, il sera plus rapide de parcourir les logs à la recherche de séquences d'attaques minimales pour confirmer ou infirmer l'hypothèse d'une attaque informatique. Le principe est le suivant : Un mot $w = a_1a_2a_3 \dots a_n$ de L_{sys} est un mot interdit minimal si et seulement si :

$$\begin{cases} w \text{ est interdit : } w \notin Fact(L_{sys}) \\ \text{les radicaux non-triviaux de } w \text{ (ni le mot vide, ni } w \text{ lui-même) ne sont pas interdits :} \\ a_1a_2 \dots a_{n-1} \in Fact(L_{sys}) \text{ et } a_2 \dots a_n \in Fact(L_{sys}) \end{cases}$$

La génération du langage reconnaissant l'ensemble des séquences minimales interdites a été décrite dans [BCM+03] : Soit L un langage sur un alphabet Σ tel que $\forall u, v \in \Sigma^* uv \in L \implies u, v \in L$. En d'autres termes, pour tout mot de L, les radicaux de cet mot sont aussi des mots de L. Dans ce cas, l'ensemble des mots

interdits minimaux de L, $MF(L)$ est défini comme suit :

$$MF(L) = \Sigma L \cap L\Sigma \cap \overline{L}$$

Si nous reprenons le système présenté en 4.5, l'automate décrivant le langage $L_{minattacks} = MF(Fact(L_{sys}))$ est donné en figure 4.10.

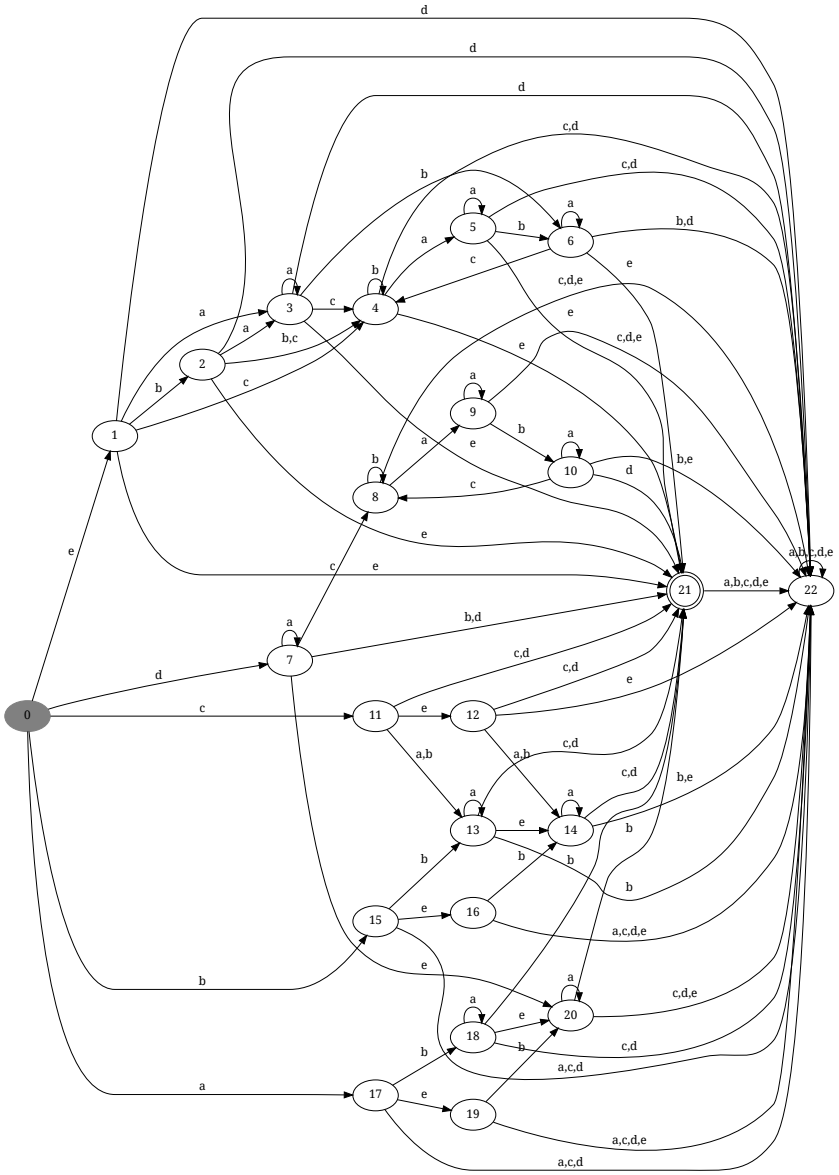


FIGURE 4.10 – Automate des séquences d'attaques minimales

4.6 Conclusion

Dans ce chapitre, nous avons présenté un système de détection d'intrusions pour les réseaux automobiles embarqués. Comme il semble difficile aujourd'hui de définir et généraliser des signatures d'attaques contre les réseaux automobiles, notamment en raison de leur grande diversité, nous avons opté pour une approche comportementale partant des spécifications d'un véhicule. Nous avons identifié quatre types de comportements suspects que nous souhaitons pouvoir détecter. À cette fin, nous proposons une détection reposant sur deux mécanismes distincts. Tout d'abord un ensemble de règles visant à contrôler que chaque trame respecte bien les spécifications du système. Par suite, notamment pour les cas où l'attaquant a un contrôle plus avancé du système et peut remplacer intégralement des messages légitimes par les siens, nous avons proposé une approche basée sur les langages formels dont l'objectif est de déterminer si un message fait partie d'une séquence interdite ou doit encore être potentiellement considéré comme légitime. Ces séquences d'attaques constituent un langage formé à partir des automates modélisant le comportement d'un point de vue du réseau des ECU constituant les systèmes surveillés. Par suite, pour éviter une possible explosion du nombre d'états des automates permettant d'identifier ces séquences, nous avons proposé un algorithme reposant sur un automate de taille bornée. Cette méthode est cependant plus coûteuse en terme de complexité temporelle. Enfin, nous avons montré comment il était également possible de générer l'ensemble des séquences d'attaques minimales d'un langage régulier tel que ceux que nous manipulons. Cela peut s'avérer utile lors d'analyses *forensics* d'enregistrements réseaux à la recherche d'éventuels signes d'une attaque.

L'évaluation expérimentale de notre IDS dans le cadre d'un réseau CAN ainsi que les limites de cette approche font l'objet du chapitre suivant.

Chapitre 5

Expérimentations

Sommaire

Ce chapitre est consacré à l'application des méthodes décrites dans le chapitre 4 sur deux cas d'utilisation pratiques. Il s'agence comme suit. Tout d'abord, nous présentons le protocole expérimental. Nous y décrivons le système que nous évaluons ainsi que les paramètres que nous cherchons à mesurer. Ensuite, nous présentons une implémentation générale des différents mécanismes nécessaires au fonctionnement de notre système de détection d'intrusions. La section suivante est dédiée à l'application de ces mécanismes sur deux exemples de systèmes : le contrôle des feux et le régulateur de vitesse. Pour des raisons de confidentialité, ces systèmes sont décrits dans ce chapitre sous un aspect simplifié, tout en permettant d'illustrer les principes décrits dans les chapitres précédents. Ils ne reflètent pas concrètement le fonctionnement réel d'un véhicule. La section se conclut sur une remarque concernant les protocoles de diagnostic. Nous présentons enfin l'évaluation de notre système, en considérant ses performances et ses limitations vis-à-vis des exemples précédemment détaillés.

5.1 Protocole expérimental

Dans cette section, nous présentons une vue d'ensemble de nos expérimentations. Nous y décrivons le système à évaluer, le matériel et les données utilisées ainsi que les différents paramètres selon lesquels nous évaluons notre système.

5.1.1 Vue d'ensemble

Afin d'obtenir une validation très poussée de notre système, il faut pouvoir disposer pendant plusieurs semaines, voire plusieurs mois du même véhicule de façon à intégrer notre IDS dans son architecture embarquée, mener une campagne d'attaques et tester les réactions de cet IDS. Comme il était difficile de rassembler de telles conditions dans le cadre d'une thèse, réaliser une telle évaluation dans le temps imparti n'a pas été possible. Néanmoins, nous avons pu enregistrer en amont le trafic réseau d'une voiture correspondant à des utilisations « standard » (c'est-à-dire sans aucune injection de messages malveillants sur le réseau) de celle-ci. Nous avons alors simulé l'occurrence d'attaques en modifiant ces enregistrements (en insérant de nouvelles trames, pour simuler les cas où l'attaquant injecte du trafic supplémentaire dans le réseau, ou en modifiant des trames existantes pour simuler les cas où l'attaquant contrôle les émissions d'un ou plusieurs ECU). Ces enregistrements modifiés peuvent ensuite être utilisés pour tester notre système de détection. Le système que nous évaluons lors de nos expérimentations est ainsi présenté en figure 5.1. À partir des spécifications décrivant l'architecture du réseau embarqué dans la voiture, nous générons les automates nécessaires aux analyses de cohérence ainsi que les règles de contrôle pour les trames impliquées dans l'évolution de ces automates. En ce qui concerne les autres trames enregistrées, comme nous ne possédons pas nécessairement les spécifications décrivant leur utilisation, nous nous basons sur les enregistrements pour déterminer leurs caractéristiques. Les règles et automates ainsi créés sont passés à notre système de détection d'intrusion qui les utilise alors pour analyser les enregistrements que nous avons modifiés pour y inclure des attaques et signale alors toutes les anomalies qu'il y détecte.

5.1.2 Données expérimentales

Les essais ont été menés sur un ordinateur de bureau équipé d'un processeur *Core i7-3720QM 2.60GHz* de Intel. Afin de nous rapprocher au plus d'un matériel pouvant être embarqué dans un véhicule récent, nous utilisons pour nos mesures un seul cœur du processeur que nous cadençons à 1,2GHz.

Nous possédons 102 fichiers contenant des enregistrements correspondant à des scénarios d'utilisations standards d'une voiture. Le nombre de trames contenues dans ces enregistrements varie de 3000 à 400000 trames environ, pour des durées d'enregistrement allant de 2 secondes à 3 minutes. Au total, ces fichiers représentent 39 minutes d'enregistrement cumulées.

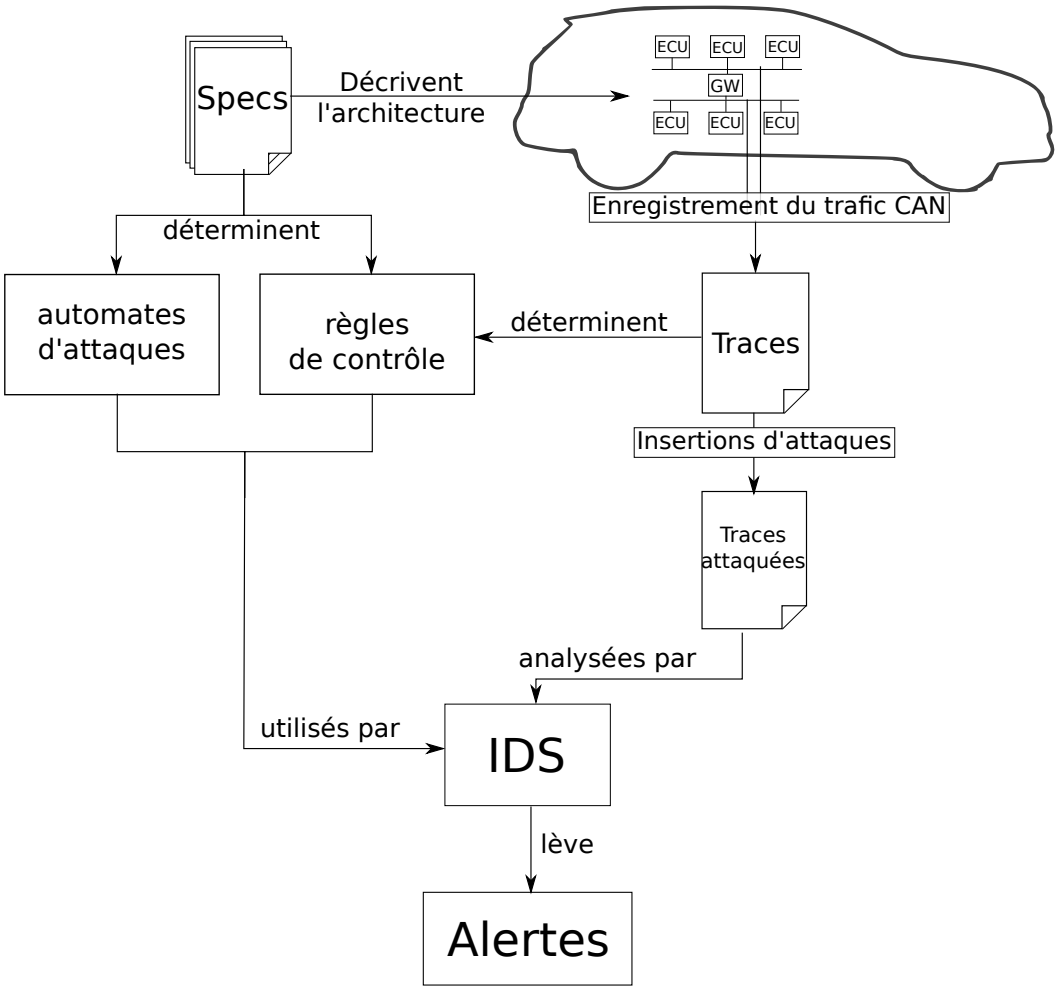


FIGURE 5.1 – Présentation du dispositif expérimental

5.1.3 Évaluation du système

L'évaluation des performances de notre système repose sur deux aspects :

- La couverture, c'est-à-dire la quantité d'attaques détectées et/ou de faux positifs. Or, il n'existe pas à l'heure actuelle suffisamment d'exemples concrets d'attaques réelles contre des réseaux CAN embarqués qui pourraient fournir une base de scénarios d'attaques généralisables que l'on pourrait tester. Ainsi, comme nous sommes ici à l'origine de toutes les attaques à tester et que celles-ci découlent des mêmes spécifications que nous avons utilisées pour concevoir notre IDS, il devient difficile de déterminer précisément la couverture de notre système par rapport à des attaques réelles. Nous mènerons les réflexions concernant la couverture selon les exemples présentés par la suite, en déterminant des catégories d'attaques qu'il est possible de détecter ou non avec l'implémentation que nous proposons.
- La vitesse de détection, que nous évaluons en mesurant le temps mis pour analyser chaque trame. Pour ce faire, nous mesurons le temps nécessaire à notre prototype pour analyser un enregistrement d'une taille donnée en faisant varier le nombre d'analyseurs utilisés, leur type ainsi que la taille des automates qu'ils utilisent. Le temps total pris pour analyser un fichier entier nous donne ainsi la durée moyenne d'analyse d'une trame. De plus, nous mesurons également les durées observées pour les différentes étapes du processus de détection d'intrusion, soit :
  - l'interprétation des données
  - la vérification des règles de contrôle
  - l'ensemble des tests de cohérence appliqués à une trame
  - la totalité de l'analyse d'une trame

5.2 Mise en œuvre

Concernant l'implémentation de la détection, rappelons tout d'abord les principales étapes du processus de détection d'intrusion que nous mettons en place. Tout d'abord, il faut récupérer les données à analyser, dans notre cas, des trames CAN circulant sur le ou les bus que l'on surveille. Comme nous l'avons vu dans le chapitre 1, une trame CAN est constituée de plusieurs champs, et le champ de données lui-même peut contenir plusieurs signaux distincts portant des informations parfois indépendantes. Dans le cas d'une trame CAN, cette étape va donc consister à séparer les différents champs dont nous aurons besoin et à isoler les différents signaux contenus dans le champ de données.

En fonction du type de la trame, déterminé par son identifiant, les analyses sont alors effectuées. Celles-ci sont de deux types : une vérification de règles et un test de cohérence basé sur des automates.

Enfin, les alertes levées par ces analyses peuvent être corrélées sur un intervalle de temps donné afin d'évaluer la criticité de la menace en cours.

Nous présentons par la suite l'implémentation de chacune de ces étapes. Le développement du prototype a été fait en utilisant le langage Python.

5.2.1 Source des données

Un extrait d'un enregistrement est présenté en figure 5.2. Chaque ligne correspond à la transmission d'une trame sur le réseau. Les colonnes décrivent respectivement :

1. l'horodatage de la trame (en secondes) ;
2. le numéro du bus sur lequel la trame a été transmise (ici, 1 ou 2) ;
3. l'identifiant de la trame, au format hexadécimal ;
4. la taille du champ de données d'après l'en-tête de contrôle ;
5. le contenu du champ de données, les valeurs sont regroupées par octet et représentées en notation hexadécimale.

Horodatage	Bus	Id	Taille	Data
2.866249	1	17E	8	FF FF FF 00 FF 40 30 FF
2.870143	1	12E	8	CC 7F FF 80 10 FF FF 00
2.870961	1	186	7	55 F0 34 63 45 00 20
2.871195	1	42E	8	68 78 D8 5E E0 00 00 00
2.871433	1	17A	8	FF FF FF BB 00 C0 33 01
2.871625	1	189	5	FF FF FF FF F0
2.871957	1	C6	8	7F EB 7F FF 80 22 AE C7
2.875697	1	18A	6	FF F0 35 86 40 00
2.875942	1	1F6	8	1E 20 00 31 00 FF 00 FF
2.876188	1	17E	8	FF FF FF 00 FF 40 30 FF
2.880136	1	12E	8	CC 7F FE 80 10 FF FF 00
2.880362	1	242	7	00 00 FF EF FE 00 0C
2.880580	1	186	7	55 F0 34 43 44 00 20

FIGURE 5.2 – Extrait d'un enregistrement de trafic CAN

Pour interpréter le contenu du champ de données, nous possédons un fichier décrivant la répartition des signaux au sein de chaque trame ainsi que leur signification. Chaque signal correspond en fait à une certaine portion des bits du champ de données et sa valeur est déterminée par la valeur de ces bits. La figure 5.3 illustre une possible répartition de 5 signaux dans un champ de données de 3 octets. Dans cet exemple, le signal S_1 contient ainsi la séquence de bits 10001. Il est à noter que l'intégralité du champ de données n'est pas nécessairement utilisée pour transmettre les signaux. Ainsi, dans notre exemple, deux bits du deuxième octet ne sont pas pris en compte.

Nous définissons ainsi une classe Trame, dont les attributs correspondent à tous les éléments décrits précédemment, soit :

– l'identifiant de la trame ;

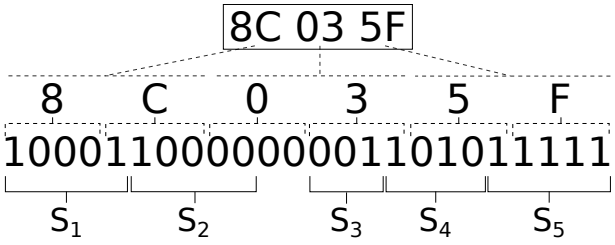


FIGURE 5.3 – Exemple de répartition de 5 signaux dans 3 octets du champ de données

- l'horodatage ;
- le bus sur lequel la trame a été transmise ;
- le champ de données brut ;
- la liste des différents signaux qu'elle contient et leurs valeurs respectives.

5.2.2 Règles de contrôle

La première étape de la détection consiste à vérifier que la trame analysée respecte bien un certain nombre de règles définies par les spécifications du réseau. Ces règles, comme les automates décrits dans le chapitre 4, sont normalement générées à partir des spécifications du système. Cependant, nous n'avons pas eu accès aux spécifications correspondant à l'ensemble des calculateurs de la voiture pour laquelle nous avons obtenu des enregistrements. Par conséquent, nous avons reconstitué l'information à partir des enregistrements. Afin d'obtenir des règles de contrôle pour toutes les trames présentes dans ces enregistrements, nous avons développé l'algorithme présenté en figure 5.4. Son principe est le suivant. Pour chaque trame comprise dans un enregistrement dont nous n'avons encore jamais vu l'identifiant, nous récupérons son identifiant, la taille de son champ de données ainsi que le bus sur lequel elle a été transmise, qui nous permettent de définir les règles de contrôle correspondante, et enregistrons également son horodatage. Ensuite, si une trame portant à nouveau cet identifiant est lue, nous n'enregistrons que son horodatage, les autres paramètres ne changeant pas. Finalement, une fois l'enregistrement parcouru, nous calculons la fréquence moyenne d'apparition de chaque trame, que nous arrondissons à la milliseconde.

Nous obtenons ainsi quatre critères à contrôler lors de l'analyse de chaque trame :

- Son identifiant est-il connu ?
- Est-elle émise sur le bon bus ?
- Le champ de données fait-il la bonne taille ?
- Si la trame est émise périodiquement, sa fréquence d'émission est-elle correcte ? Ce critère n'est vérifié qu'à partir de l'observation d'une deuxième trame portant le même identifiant.

Une fois ces analyses effectuées, la trame est alors soumise à un ou plusieurs contrôles de cohérences, en fonction du nombre de systèmes définis qu'elle concerne.

```
def learn(liste_trames):
  identifiers={} # dictionnaire dont les clés sont les identifiants
  # des trames et les valeurs sont les paramètres a vérifier
  freqs={} # utilisé pour calculer les fréquences d'émission de chaque
  # trame
  for trame in liste_trames:
    if trame.ident not in identifiers.keys():
      identifiers[trame.ident]={
        "rules":{"size":datasize, "bus":bus,"freq":0}
      }
      freqs[trame.ident]=[]
  freqs[trame.ident].append(timestamp)

  for i in freqs.keys():
    meanfreq=0
    if len(freqs[i]) > 1:
    # calcul de la fréquence moyenne d'une trame
    # arrondie a la milliseconde
      for k in range(len(freqs[i])-1):
        meanfreq = (meanfreq*k + freqs[i][k+1] - freqs[i][k])/(k+1)
    identifiers[i]["rules"]["freq"]=round(meanfreq,3)
  return identifiers
```

FIGURE 5.4 – Génération de règles

5.2.3 Contrôles de cohérence

Comme nous l'avons présenté dans la section 4.3 du chapitre précédent, le contrôle de la cohérence d'une trame est effectué grâce à un ou plusieurs automates finis construits à partir d'un langage rationnel qui représente les possibles séquences de messages envoyés sur le réseau par un système du véhicule. Or, les symboles qui étiquettent les transitions de ces automates ne correspondent pas nécessairement directement à une donnée contenue dans une trame (telle que l'identifiant). Ils peuvent ainsi correspondre à des évènements du type « la porte conducteur est ouverte » ou encore « la vitesse est supérieure à Nkm/h », qui doivent être déduits des informations contenues dans le champ de données. Avant d'effectuer les contrôles de cohérence sur une trame, il convient donc de générer les symboles correspondants.

Génération d'un symbole

Ces symboles sont générés par des fonctions qui prennent une trame en entrée, analysent les informations qu'elle contient (identifiant, valeurs des signaux) et renvoient un symbole en sortie. On considère ainsi que pour un système, une trame donnée ne génère qu'un symbole à la fois. Cependant, si plusieurs systèmes sont concernés par cette trame, celle-ci peut alors être à l'origine de symboles différents pour chacun de ces systèmes.

La génération d'un symbole peut parfois nécessiter de comparer les informations

contenues dans la trame avec une ou plusieurs variables d'état du système. Par variable d'état, nous entendons une valeur caractérisant un paramètre commun à différents éléments du système. La vitesse du véhicule constitue ainsi un exemple de variable d'état. Ces variables sont mises à jour lors de l'analyse d'une trame par certaines fonctions, en parallèle avec la génération de symbole.

La fonction présentée en figure 5.5, tirée de l'exemple du régulateur de vitesse détaillé en 5.3.2 peut ainsi générer quatre symboles différents, *CCoff*, *CCon*, *CConVinf* ou *CConVsup* à partir d'une trame contenant les signaux nommés *EtatCC* et *VitRequise*. Celle-ci utilise également une variable d'état nommée *curspeed* qui contient la vitesse actuelle du véhicule (mise à jour lors de l'analyse d'une autre trame). Dans cet exemple, nous envisageons également le cas où la vitesse à réguler demandée par le conducteur est jugée incohérente (supérieure 200km/h), ce qui se traduit par la levée d'une alerte.

```
1  def CCinfo(trameCC, analyseur):
2    if (trameCC.signals["EtatCC"] == 1):
3      reqspeed=trameCC.signals["VitRequise"]
4      if (reqspeed < 200):
5        if (reqspeed < analyseur.stateVars["curspeed"]-5.0):
6          return "CConVsup"
7        elif (reqspeed > analyseur.stateVars["curspeed"]):
8          return "CConVinf"
9        else:
10         return "CCon"
11       else:
12         #la fonction alert sert au traitement des valeurs incohérentes
13         alert("Vitesse␣incohérente")
14   else:
15     return "CCoff"
```

FIGURE 5.5 – Exemple de détermination de symboles

L'objet *analyseur*, manipulé par cette fonction, est présenté ci-après.

Méthodes d'analyse

Nous définissons une classe appelée *Analyseur*, qui possède les attributs suivants :

- Un automate qui sera utilisé lors de l'analyse.
- Un dictionnaire *dictSymboles* où chaque clé est un identifiant d'une trame concernée par l'automate et dont la valeur associée pointe vers la fonction à appeler pour générer le symbole correspondant aux informations contenues dans cette trame.
- Une liste contenant les variables d'état dont peuvent avoir besoin les fonctions générant les symboles.
- Une liste contenant l'ensemble des symboles lus par l'automate jusqu'à présent.

Pour effectuer les deux méthodes de détection décrites dans le chapitre 4 (avec un AFD décrivant les séquences interdites ou en reprenant l'automate des préfixes), nous avons créé deux classes, *AnalyseurAFD* et *AnalyseurRad*, qui héritent de *Analyseur*.

- Un *AnalyseurAFD* possède ainsi un attribut supplémentaire : l'état dans lequel se trouve actuellement l'AFD qui est utilisé pour la détection d'intrusion. Il possède une méthode *analyse* qui prend une trame en entrée, met à jour l'état actuel de l'automate. Si ce dernier est un état terminal, une alerte est levée et l'état de l'analyseur redevient l'état initial de l'automate afin de commencer une nouvelle analyse.
- Un *AnalyseurRad*, possède lui une liste qui contient l'ensemble des états possibles dans lesquels l'automate correspondant à L_{sys} peut se trouver (cf Algorithme 1). Sa méthode *analyse* reprend ainsi l'algorithme 1 décrit en 4.4.2. Là aussi, si une alerte est levée, on remet l'automate dans son état initial en réinitialisant la liste des états à l'ensemble des états de l'automate.

Ainsi, lorsqu'une trame a été jugée valide suite aux premiers contrôles, celle-ci est alors transmise aux analyseurs concernés. La liste des analyseurs concernés par une trame donnée est préalablement déterminée. Pour notre prototype en Python, celle-ci est stockée sous la forme d'un dictionnaire où les clés sont les identifiants des trames et les valeurs sont des listes d'analyseurs. Chaque analyseur recevant la trame détermine alors quelle fonction appeler pour générer le symbole correspondant grâce à son dictionnaire *dictSymboles*. Ensuite, l'analyseur tire la transition correspondant à l'état actuel de l'automate et au symbole ainsi généré pour en déduire le nouvel état du système qu'il observe.

5.2.4 Corrélation des alertes

Si, lors de chacune des analyses menées, une anomalie est détectée, il est possible de ne pas lever directement une alerte si l'on juge que cette anomalie est mineure. Pour prendre cette décision, un module est chargé de recueillir les anomalies identifiées par les modules d'analyse et de les corréler selon la méthode décrite en 4.2.2. Cependant, établir des pondérations et définir les valeurs de seuils correspondantes constitue une tâche très complexe et devra faire l'objet de travaux ultérieurs. En effet, cela nécessite de mener des analyses de risques pour chaque système surveillé puis d'établir des campagnes d'attaques et de test de l'IDS afin de pouvoir déterminer précisément quelles valeurs permettront de minimiser les occurrences de faux positifs et faux négatifs. Ainsi, l'évaluation d'un système de corrélation des alertes levées par ces mécanismes ne sera pas présentée ici. Notre objectif dans la suite de ce chapitre sera donc de déterminer l'efficacité des différents mécanismes de détection introduits dans le chapitre 4.

Nous allons désormais illustrer le fonctionnement de cette implémentation avec deux cas d'application que nous utilisons par la suite pour évaluer les performances de notre système.

5.3 Cas d'étude

Cette section est dédiée à la présentation et à la modélisation des systèmes qui nous ont servi à concevoir et évaluer notre prototype de système de détection d'intrusion. Pour chacun de ces exemples, nous décrivons ainsi le fonctionnement du système que l'on souhaite surveiller, décrivons les exemples d'attaques possibles contre un tel système, présentons les automates obtenus en appliquant l'approche décrite dans le chapitre 4 sur ces systèmes puis évaluons la capacité de notre système à détecter de telles attaques. Nous rappelons ici que les systèmes présentés constituent des versions simplifiées de systèmes réels.

5.3.1 Contrôle des feux

Notre premier exemple est extrêmement simplifié afin de nous permettre d'illustrer clairement le principe de fonctionnement de notre système. Il concerne l'allumage et l'extinction des différents feux d'une voiture.

5.3.1.1 Présentation du système

Ce système peut être décrit directement avec un seul automate, présenté en figure 5.6. Son fonctionnement est le suivant :

- Initialement, les feux sont éteints (symbole *off*).
- Lorsque le conducteur actionne la commande d'allumage des feux, il active d'abord les feux de position, ce qui entraîne l'envoi d'une trame contenant l'ordre d'allumer ces derniers. Cela correspond au symbole *pos*.
- Il peut ensuite choisir d'allumer les feux de croisement, puis les feux de route. Cela correspond respectivement aux symboles *crois* et *phare*.
- De même, pour éteindre les feux, le conducteur fait passer la commande par tous les états intermédiaires entre l'état actuel du système et l'état *off*.

Les messages commandant l'allumage ou l'extinction des feux correspondent à différentes valeurs d'un même signal Sig_{feux}, contenu dans une seule trame portant l'identifiant T_{feux} qui est émise périodiquement.

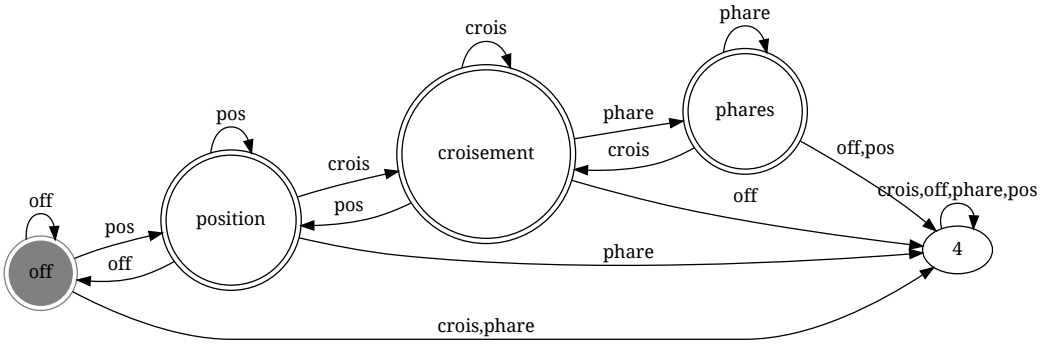


FIGURE 5.6 – Automate du contrôle des feux

5.3.1.2 Scénarios d'attaque

Le système étant simple, les possibilités d'attaque sont ici limitées. Sur un tel système, un attaquant peut donc uniquement tenter de forcer un mode d'éclairage donné. Le cas le plus dangereux, une extinction des feux alors que l'automobile roule la nuit, menace la *safety* du véhicule. Nous envisageons ici deux types d'attaques. Dans le premier cas, l'attaquant envoie des trames alors que la commande des feux continue d'émettre périodiquement des données. Dans ce cas, les feux changeront d'état à chaque fois qu'ils recevront une trame provenant du système légitime ou de l'attaquant. Si l'attaquant n'émet pas de trames avec une fréquence suffisamment élevée, il y aura alors un clignotement rapide des feux. Dans le second cas, l'attaquant peut interrompre à tout moment la transmission de données légitimes et émettre seulement les messages de son choix. Ici, les feux seront donc entièrement contrôlés par l'attaquant.

5.3.1.3 Détection des attaques

Nous présentons ici les possibilités de détection de notre système vis-à-vis des scénarios d'attaque décrits précédemment.

Règles de contrôle Théoriquement, si on se place dans le cas où l'attaquant ne fait qu'envoyer des trames supplémentaires sur le réseau, une simple vérification de la fréquence d'apparition de T_{feux} peut suffire à détecter une attaque en cours. Cependant, lors de l'analyse des enregistrements correspondant à une utilisation standard des feux, nous avons constaté que la trame T_{feux} n'était pas uniquement émise périodiquement. En effet, lorsque le conducteur actionne la commande des feux, une trame T_{feux} supplémentaire contenant la nouvelle valeur de Sig_{feux} est émise sur le bus. Ainsi, utiliser simplement la fréquence d'émission de T_{feux} s'avère insuffisant car cela entraîne alors un faux-positif à chaque fois que le conducteur actionne la commande des feux. Pour éviter cela, l'utilisation de mécanismes de corrélation des alertes comme décrit en 4.2.2 est nécessaire. Par exemple, une augmentation de fréquence sera considérée suspecte si elle se produit plus de n fois en un intervalle de temps donné (nous retrouvons ici la notion de fenêtre glissante introduite à ce sujet dans la section 4.2.2). Un attaquant connaissant la présence d'un système de détection d'intrusions utilisant ces règles sera toujours capable de perturber le système sans être détecté (toute instruction reçue par les feux étant exécutée), mais à un rythme bien moins soutenu ne mettant pas en danger les passagers du véhicule. Dans le cas où l'attaquant peut contrôler l'émission de toutes les trames T_{feux} à partir d'un instant donné, cette méthode ne permet pas de détecter ses actions.

Test de cohérence Le système étant très simple, l'utilisation d'un AFD permet d'effectuer ces tests en un minimum de temps sans pour autant demander trop de mémoire. L'AFD $A_{attacks}$ correspondant au système est représenté en figure 5.7. La

fréquence d'apparition des trames n'entre plus en jeu ici, seul l'ordre d'apparition sert de base à la détection. Les attaques alors détectables sont celles qui forcent une transition du contrôle des feux entre deux états non adjacents. Ici, cela correspond à un passage de *off* à *croisement*, de *off* à *route*, de *position* à *route* ainsi que les transitions inverses. En revanche, si un attaquant force une transition entre deux états consécutifs, celle-ci ne sera pas détectée par l'automate. Dans le cas où l'attaquant contrôle l'émission de toutes les trames T_{feux} à partir d'un instant donné, cette méthode peut détecter certaines de ses actions si celles-ci correspondent aux cas décrits ci-dessus.

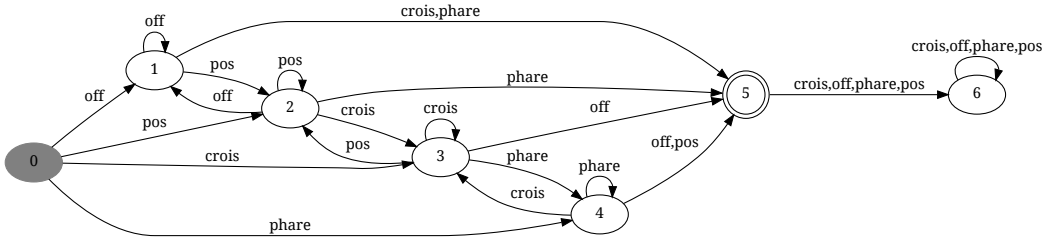


FIGURE 5.7 – AFD décrivant le langage d'attaques du contrôle des feux

Conclusion Les deux approches sont complémentaires puisqu'elles permettent de détecter deux cas différents. Cependant, un attaquant pourra toujours parvenir à perturber faiblement le système sans être détecté, mais cela n'affectera plus la *safety* du système.

5.3.2 Régulateur de vitesse

Le cas du régulateur de vitesse est plus complexe. En effet, cette fonction fait interagir trois sous-systèmes. Ce cas nous permet ainsi d'illustrer l'intégralité de la démarche présentée dans le chapitre 4 : après avoir modélisé chaque sous-système par un automate, nous effectuons la composition parallèle synchronisée de ces automates pour représenter l'ensemble des évolutions possibles de la régulation de vitesse. C'est alors à partir de cette composition que nous générerons les séquences d'attaques à détecter.

5.3.2.1 Présentation du système

Le contrôleur du régulateur de vitesse est présenté en figure 5.8. Initialement arrêtée (*CCoff*), la fonction régulateur de vitesse ne peut être activée que lorsque le véhicule a dépassé une certaine vitesse *Vok*. Dès lors, si le conducteur actionne la fonction régulateur de vitesse à partir de cet instant, une trame contenant l'ordre de mettre en route la régulation de vitesse (*CCon*) est émise. Cette trame contient également un signal donnant la vitesse à laquelle le régulateur de vitesse doit maintenir le véhicule. À partir de cette information et de la vitesse actuelle du véhicule, trois cas sont possibles : la vitesse actuelle est inférieure (*CConVinf*), supérieure

(*CConVsup*) ou égale (*CCon*) à la vitesse demandée. Si la vitesse du véhicule est inférieure, une instruction demandant une accélération (*accel*) est émise. À l'inverse, si elle est supérieure, un ralentissement (*ralent*) est demandé. Finalement, la régulation de vitesse peut être arrêtée à tout moment si le conducteur coupe manuellement la fonction, appuie sur la pédale de frein (*fc*) ou si la vitesse du véhicule redescend sous la valeur *Vok* (*Vnok*).

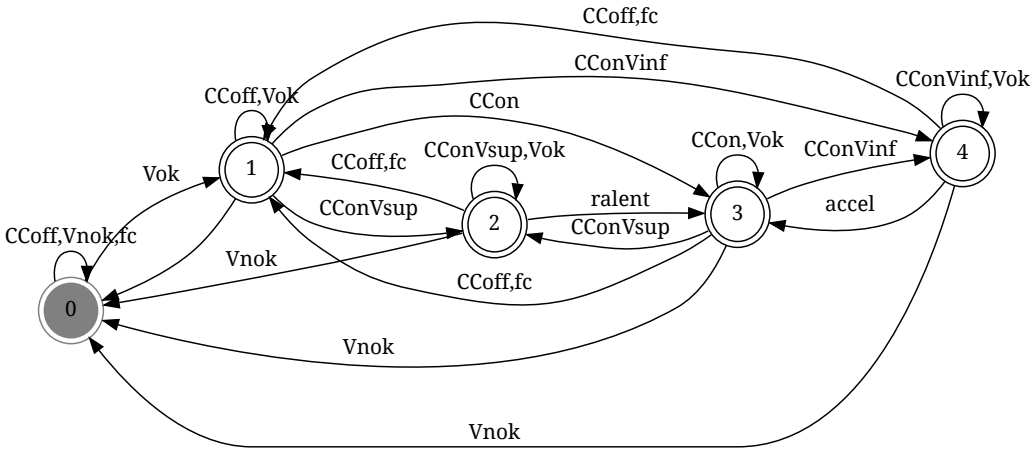


FIGURE 5.8 – Fonctionnement simplifié d'un régulateur de vitesse

Le capteur donnant l'état de la pédale de frein est présenté en figure 5.9. Celui-ci émet une trame dont un signal atteste de l'état de la pédale de frein. Celle-ci peut tout d'abord être relâchée (*f0*). Ce capteur émettant des trames à une fréquence très élevée, pour que la pédale soit confirmée comme étant appuyée, une pression doit être détectée durant plusieurs cycles : le capteur doit signaler ainsi une pression détectée (*fa*) durant 2 émissions avant de confirmer que la pédale est appuyée (*fc*).

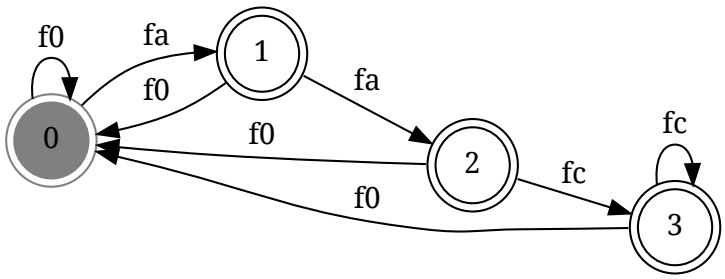


FIGURE 5.9 – Fonctionnement simplifié du capteur de la pédale de frein

La vitesse du véhicule est considérée ici comme un système à part entière. Le signal vitesse, contrairement aux signaux évoqués précédemment, est une donnée continue. Il n'est donc pas possible de représenter via un automate fini tous les

changements de valeurs de celle-ci. Nous ne considérons donc ici que des valeurs paliers : vitesse nulle ($V0$), vitesse trop basse pour le régulateur de vitesse ($Vnok$), vitesse autorisée pour le régulateur de vitesse (Vok). L'automate décrivant l'évolution de la vitesse est présenté en figure 5.10. Initialement, le véhicule est à l'arrêt, la vitesse est donc nulle et le levier est en position parking. Pour commencer à se déplacer, le conducteur place le levier en position *drive* et accélère. La vitesse peut alors passer de *V0* à *Vnok*, puis de *Vnok* à *Vok* ou inversement.

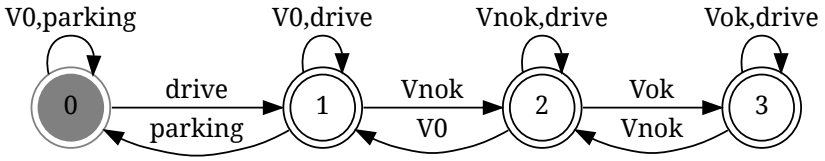


FIGURE 5.10 – Évolution de l'information vitesse

La composition de ces trois systèmes, présentée en figure 5.11, nous donne alors une description des évolutions possibles de l'ensemble des systèmes impliqués dans la régulation de la vitesse.

5.3.2.2 Scénarios d'attaque

Parmi les attaques envisageables, les cas qui nous intéressent ici sont ceux où l'attaquant va chercher à contrôler l'accélération du véhicule, impactant alors les contraintes de *safety* liées au régulateur de vitesse. Pour ce faire, il peut tout d'abord émettre des trames contenant le signal *accel* ou *ralent* normalement émises par le régulateur de vitesse. Il peut également émettre une information vitesse fausse pour perturber la prise de décision du régulateur de vitesse. D'autres attaques peuvent viser à couper le régulateur de vitesse lorsque celui-ci est en fonctionnement en envoyant des trames qui entraînent la génération des symboles *Vnok*, *CCoff* ou *fc*.

5.3.2.3 Détection des attaques

Nous présentons ici les possibilités de détection de notre système vis-à-vis des scénarios d'attaque décrits précédemment.

Règles de contrôle Tout d'abord, précisons que toutes les trames impliquées dans le fonctionnement de ces systèmes sont purement périodiques. Ainsi, dans le cas où un attaquant ne peut pas interrompre le trafic légitime avant d'émettre ses messages, toute augmentation de la fréquence d'apparition d'un message peut être considérée comme suspecte et permet donc de détecter une attaque en cours. Cependant, si l'on ne souhaite pas établir des règles de corrélation d'alertes trop spécifiques (ce qui irait à l'encontre de la contrainte de compatibilité) et donc attendre de constater une augmentation de la fréquence sur une durée de temps plus étendue avant de lever une alerte, il peut être possible de détecter une intrusion plus tôt grâce aux tests de cohérence.

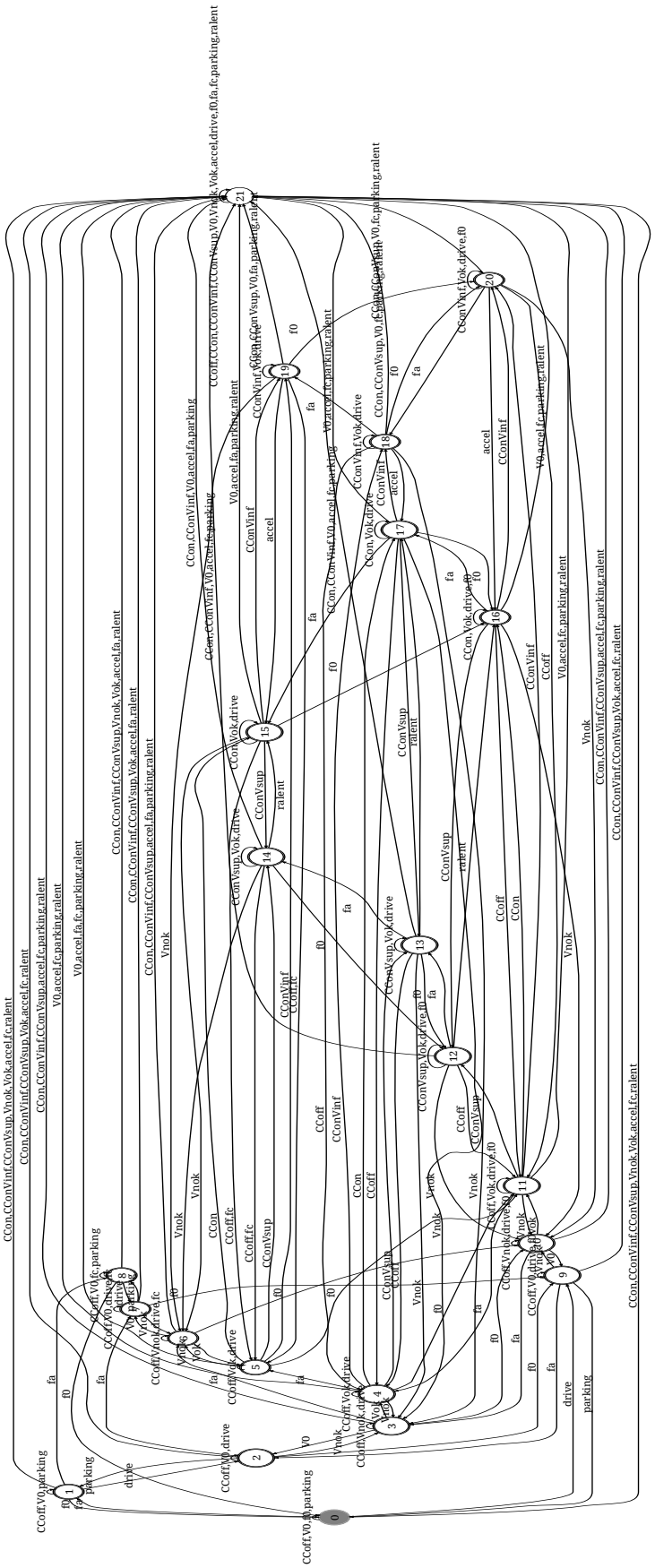


FIGURE 5.11 – Composition synchronisée des automates 5.8, 5.9 et 5.10

Test de cohérence Considérons maintenant les cas où l'attaquant peut choisir d'être le seul à émettre certaines trames. Par exemple, s'il tente de faire accélérer ou ralentir le véhicule en émettant des trames contenant les signaux *accel* ou *ralent* lorsque ce n'est pas nécessaire, nous détecterons cette anomalie avec une seule trame. De même s'il n'envoie qu'une trame *fc* pour couper le régulateur de vitesse. En revanche, si l'attaquant connaît parfaitement le fonctionnement du système qu'il contrôle, et que ce système fonctionne indépendamment du reste du véhicule (il ne réagit pas à certaines trames transmises sur le réseau et les signaux qu'il émet ne sont pas redondants ni ne peuvent être recoupés avec d'autres informations transmises sur le réseau), il peut alors passer outre nos mécanismes de détection en émettant la bonne séquence de trames. C'est le cas du capteur de la pédale de frein dans notre exemple. En effet, celui-ci représente notre unique source d'information sur l'état de la pédale de frein et les trames qu'il émet ne sont pas conditionnées par des messages envoyés par d'autres calculateurs sur le réseau. En l'état, un attaquant, capable de contrôler les trames émises par ce capteur, envoyant une séquence de freinage valide (*fa, fa, fc*) déclencherait alors un arrêt du régulateur de vitesse sans être détecté. De même, nous ne présentons ici qu'une unique source donnant la vitesse du véhicule, ce qui permettrait aussi à un attaquant la contrôlant de perturber le régulateur de vitesse sans être détecté. Or, si l'on envisage un champ d'analyse plus large, il peut exister d'autres sources (un capteur donnant la vitesse de chaque roue ou encore le GPS) pour permettre de confronter plusieurs valeurs de la vitesse entre elles, ce qui est un procédé déjà employé par certains calculateurs dans une optique de sécurité-innocuité. Pour information, l'automate décrivant le langage d'attaques du régulateur de vitesse (54 états) est donné en figure 5.12.

Conclusion Au vu de la relative simplicité du système de régulation de la vitesse que nous avons présenté ici, il reste possible pour un attaquant ayant une connaissance approfondie de son fonctionnement et possédant un niveau avancé de contrôle sur certains équipements de mener une attaque sans être détecté. Cependant, un tel cas constitue une situation extrêmement improbable. Dans l'ensemble, notre méthode reste capable de détecter des tentatives de prise de contrôle de l'accélération du véhicule ou d'arrêt du régulateur correspondant à des scénarios d'attaques plus probables. De plus, dans le cas d'un véhicule réel, possédant donc des systèmes plus complexes que ceux présentés ici, une plus grande diversité de messages entrent en jeu, ce qui augmente donc les possibilités de corrélation qui peuvent être effectuées.

5.3.3 Remarque : cas du diagnostic

Dans ce chapitre, nous nous concentrons sur les trames circulant sur le bus lors d'un usage « standard » d'un véhicule mais nous n'avons pas traité le cas des sessions de diagnostic. N'ayant pas eu l'opportunité d'implémenter les diverses possibilités offertes par les protocoles de diagnostic, nous avons simplement traité leur existence en nous basant sur les spécifications précisant qu'une session de diagnostic ne pouvait se produire que lorsque le véhicule se trouvait à l'arrêt. La figure 5.13 illustre

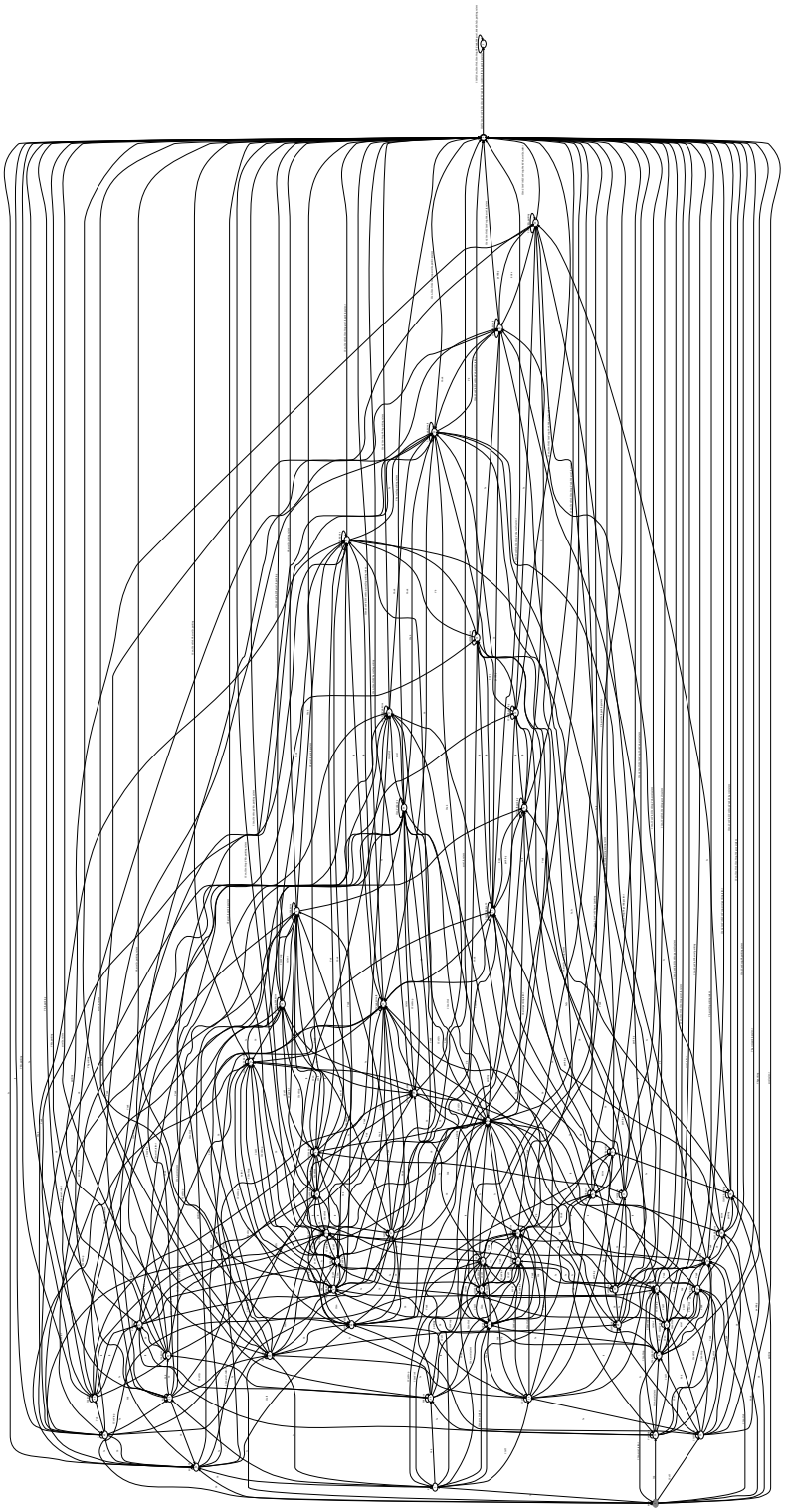


FIGURE 5.12 – Automate décrivant le langage d'attaques du régulateur de vitesse

ainsi un modèle du contrôle des feux prenant en compte le diagnostic. Concrètement, cela revient à synchroniser l'automate du contrôle des feux (figure 5.6) avec l'automate représenté en figure 5.14. Ce dernier décrit le fait qu'une session de diagnostic ne peut commencer (*startD*), se poursuivre (*diag*) puis se terminer (*endD*) que si la vitesse est nulle (*V0*). Nous pouvons donc détecter les cas où un attaquant tenterait de perturber un calculateur via des fonctions de diagnostic lorsque le véhicule attaqué se déplace. Des exemples de possibilités offertes par une telle situation sur un véhicule non protégé ont été présentés par Miller et Valasek dans [MV13].

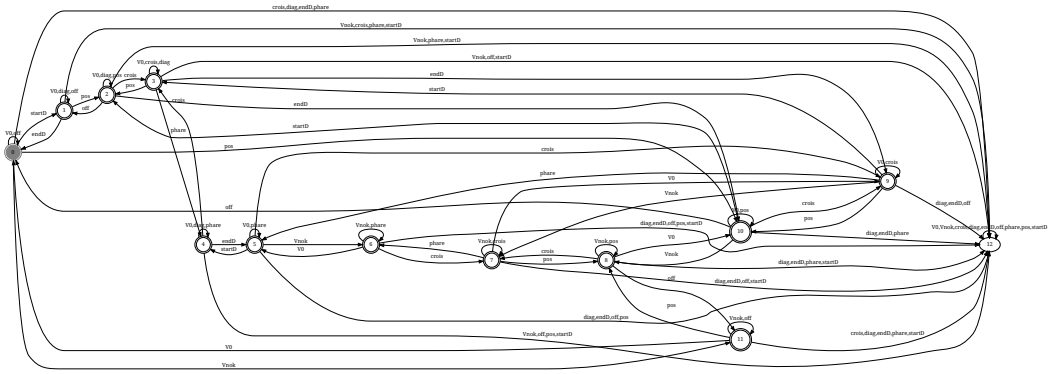


FIGURE 5.13 – Automate des feux avec prise en compte du diagnostic

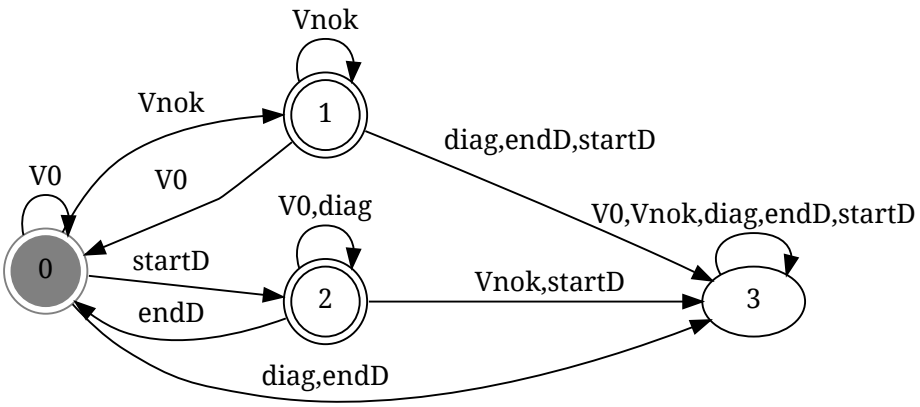


FIGURE 5.14 – Modélisation simple d'une session de diagnostic

5.4 Évaluation

Cette section est consacrée à l'évaluation de notre système. Nous nous intéressons tout d'abord aux performances des méthodes que nous avons décrites dans le chapitre précédent, puis discutons de la couverture de notre système.

5.4.1 Performances

Pour chacun des cas présentés par la suite, les résultats ont été obtenus en analysant 100 fois un fichier d'enregistrement qui contient 310440 trames, correspondant à 205 secondes de trafic CAN. En moyenne, nous avons donc 1514 trames par seconde à analyser. Le système considéré ici est le régulateur de vitesse présenté précédemment. Ainsi, seules les trames impliquées dans le fonctionnement de ce système subissent des contrôles de cohérence.

En revanche, toutes les trames sont soumises aux deux étapes précédentes, à savoir l'interprétation des données et l'application des règles de contrôle. Concernant ces deux étapes, les résultats obtenus sont présentés en table 5.1.

Étape	Moyenne	Min	Max
Interprétation	$32\mu s$	$8\mu s$	$517\mu s$
Règles	$33\mu s$	$5\mu s$	$110\mu s$

TABLE 5.1 – Temps d'exécution des mécanismes d'interprétation des données et de vérification de règles de contrôle

Les fortes variations observées dans l'interprétation sont dues à la diversité des informations pouvant être contenues dans une trame. En effet, le nombre d'opérations à effectuer sur le champ de données d'une trame peut varier en fonction de la taille du champ de données, du nombre de signaux qu'il contient et des éventuelles opérations à effectuer sur les signaux binaires pour en déduire leur valeur réelle. En effet, le champ de données d'une trame peut être de petite taille (8 ou 16 bits) et ne contenir qu'un unique signal, ce qui entraîne un traitement très rapide. À l'opposé, certaines trames peuvent contenir plusieurs dizaines de signaux stockés sur la totalité des 64 bits que peut occuper le champ de données (dans les enregistrements que nous utilisons, un type de trame contient ainsi plus de 30 signaux). Concernant les règles, chaque trame subit ici le même nombre de vérifications. Les variations entre minimum et maximum sont dues aux cas où une anomalie est détectée, ce qui entraîne la remontée d'une alerte.

Intéressons-nous maintenant aux contrôles de cohérence. Si nous considérons que notre IDS ne surveille que le régulateur de vitesse, nous obtenons les résultats présentés en table 5.2, respectivement pour les méthodes utilisant $A_{attacks}$ et A_{right} décrites dans le chapitre 4.

Méthode	Nombre d'états	Temps total	Moyenne trame	Min trame	Max trame	Moyenne cohérence	Min cohérence	Max cohérence
$A_{attacks}$	54	46,628s	150μs	34μs	600μs	12μs	10μs	46 μs
A_{right}	21	47,162s	152μs	34μs	610μs	38μs	33μs	176 μs

TABLE 5.2 – Temps nécessaire à l'IDS pour traiter 310440 trames

Nous retrouvons bien le fait que la méthode utilisant l'AFD $A_{attacks}$ effectue le contrôle de cohérence en temps constant : la moyenne et le minimum sont très proches, le maximum correspondant aux cas où une anomalie a été détectée. La

méthode se basant sur A_{right} nécessite bien un automate plus petit mais requiert plus de temps (même si à cette échelle les gains sont négligeables).

Le trafic CAN que nous observons correspondant environ à 1514 trames par seconde, il s'écoule donc en moyenne $660\mu s$ entre chaque trame transmise. Par conséquent, les durées mesurées lors de l'analyse nous permettent bien de surveiller le trafic en temps réel puisqu'aussi bien les vérifications de règles que les tests de cohérence se font respectivement en 110 et $176\mu s$ dans le pire des cas. En revanche, des durées proches de ce seuil ont été observées lorsque l'on considère la durée totale de traitement d'une trame. Celles-ci sont majoritairement dues à l'interprétation des signaux d'une trame. Des optimisations de cette procédure permettraient certainement de réduire sa durée.

Pour représenter un système plus important, nous avons calculé le produit synchronisé de tous les systèmes présentés dans ce chapitre (régulateur de vitesse, contrôle des feux et diagnostic). L'automate réduit résultant de l'opération A_{global} possède 108 états. L'automate $A_{attacks}$ correspondant en possède 366. Les analyses utilisant $A_{attacks}$ se faisant en temps constant, nous ne présentons dans le tableau 5.3 que les durées observées en utilisant A_{right}. Rappelons qu'avec cette méthode, la durée maximale de l'analyse de cohérence correspond au début de cette analyse, c'est-à-dire lorsque l'ensemble S_{right} contient tous les états de A_{right}. Or, comme nous l'avons vu en section 4.4.3, ce nombre d'états diminue fortement dès les premières trames analysées. Nous retrouvons ce fait ici avec une durée moyenne d'analyse de $49\mu s$ qui contraste fortement avec le maximum observé de $779\mu s$.

Méthode	Nombre d'états	Temps total	Moyenne trame	Min trame	Max trame	Moyenne cohérence	Min cohérence	Max cohérence
A_{right}	108	47,369s	153μs	34μs	903μs	49μs	20μs	779 μs

TABLE 5.3 – Temps nécessaire à l'IDS pour traiter 310440 trames en utilisant A_{global}

Ici, le pire cas mesuré pour une analyse de cohérence est donc supérieur au seuil des $660\mu s$. Il est alors probable que des retards se produisent ponctuellement lors de l'analyse. Cependant le temps moyen mesuré reste lui bien inférieur. La majeure partie des analyses n'est donc pas fortement impactée par l'utilisation d'un automate plus complexe.

Ainsi, lors de la modélisation de l'intégralité des calculateurs présents sur un réseau embarqué, il sera intéressant d'établir précisément quels systèmes doivent être synchronisés entre eux. En effet, si deux systèmes de taille importante ne sont pas corrélés, il devient peut-être plus avantageux d'effectuer deux analyses séparées avec deux automates de taille inférieure plutôt qu'une seule analyse utilisant un automate plus complexe. En effet, cette dernière nécessiterait d'utiliser un ensemble S_{right} beaucoup plus grand et allongerait fortement la durée d'analyse dans les pires cas.

Les expérimentations ayant été menées sur un prototype développé en python et exécuté sur ordinateur de bureau, les résultats obtenus ne sont pas directement représentatifs d'un calculateur embarqué en situation réelle. Cependant, les tendances

observées permettent de valider notre démarche, d'autant plus qu'une meilleure optimisation peut permettre d'obtenir des gains de temps non négligeables. Ainsi, une simple compilation de notre prototype effectuée par l'intermédiaire du langage *Cython*[1], qui nous permet de générer automatiquement du code C correspondant à notre implémentation en Python, offre déjà des gains de performance de l'ordre de 30%.

5.4.2 Couverture

Les automates utilisés par notre solution sont basés sur les spécifications du réseau et des ECU. Ainsi, lorsque nous avons effectué une analyse sur les enregistrements d'origine, nos tests de cohérence n'ont renvoyé aucun faux positif. En revanche, si les règles de contrôle ne définissent pas assez précisément les paramètres à vérifier, des faux positifs peuvent se produire lors de leur vérification. C'est par exemple ce que nous avons constaté sur le cas du contrôle des feux où des trames intermédiaires légitimes pouvaient être émises en plus des trames périodiques. Les solutions pour éviter ce problème consistent en une implémentation d'un mécanisme de corrélation des alertes tenant compte des spécificités de certaines trames, ce qui peut néanmoins laisser une certaine marge de manœuvre à un attaquant ou en s'assurant qu'un test de cohérence appliqué par la suite sera capable de détecter une anomalie dans la séquence de trames observée.

Cependant, il peut arriver qu'une partie du système réel ne respecte pas strictement les spécifications (comme cela a été observé dans [KCR+10]), par exemple en incluant des fonctionnalités non documentées. Des déviations par rapport au modèle peuvent se produire. Dans de tels cas, notre système peut alors considérer qu'une attaque est en cours alors qu'il observe en réalité l'une de ces déviations. Une solution possible à ce problème serait de générer (ou d'assister la génération de) l'automate A_{sys} via des algorithmes d'apprentissage. Cependant, cela nous ramène alors aux problématiques liées à la détection d'anomalie via des modèles statistiques (cf. 3.4.1).

De plus, certaines attaques peuvent n'être jamais détectées si notre observation n'a pas pu analyser certaines trames. Par exemple, considérons les deux séquences suivantes (chaque lettre est un symbole) : la première, *a(xyz)*o* est légitime, alors que la seconde, *b(xyz)*o* est interdite. Si notre système a commencé son analyse après que le premier symbole de la séquence a été transmis, il ne pourra pas détecter l'attaque correspondant à la deuxième séquence. Par conséquent, si un attaquant est capable de forcer notre IDS à recommencer une analyse à zéro, il peut alors échapper à la détection. Une telle situation est envisageable si, par exemple, notre IDS est intégré à un autre ECU pouvant être redémarré via une trame donnée, ou encore si l'attaquant choisit délibérément de faire détecter par notre IDS une séquence interdite (mais aux conséquences minimes) sur le même système, lui faisant reprendre son analyse à zéro.

1. `http://cython.org/`

Par ailleurs, nous n'avons ici travaillé qu'avec des AFD « basiques ». En effet, nous dérivons nos automates des spécifications des systèmes à surveiller. Or, ceux-ci sont souvent eux-mêmes décrits sous la forme de machines à états. Ainsi, l'utilisation d'AFD permettait une mise en œuvre aisée d'une première implémentation et convenait aux scénarios que nous devions analyser. Cependant, il n'est pas à exclure que la complexité croissante des systèmes embarqués dans l'automobile mène à une diversité de scénarios ne pouvant pas être détectés par des AFD à l'expressivité limitée. Par exemple, le temps n'est pas directement pris en compte dans nos automates (nous utilisons le premier niveau de contrôles pour vérifier la fréquence d'émission de chaque trame), ceux-ci nous servant seulement à vérifier l'ordre dans lequel les données sont transmises sur le réseau. Cependant, il pourrait exister des attaques où non seulement l'ordre, mais également les intervalles de temps entre des trames distinctes importent. En l'état, nous ne sommes pas aptes à détecter de telles attaques. En fonction des scénarios envisagés, une évolution possible de notre système pourrait inclure des automates plus expressifs (voir par exemple [SEJ08]) afin d'améliorer ses capacités de détection.

Malgré les limites que nous venons d'identifier, notre IDS s'avère efficace et performant dans une majorité des cas d'attaques identifiés, c'est-à-dire les les cas où l'attaquant ne possède pas à la fois 1) une connaissance précise du comportement attendu des différents ECU qui composent le système qu'il cible, et 2) la capacité d'empêcher l'émission de certains messages légitimes pour lesquels nous ne sommes pas en mesure d'effectuer des tests de corrélation.

5.5 Conclusion

Dans ce chapitre, nous avons présenté et évalué une première implémentation de notre système de détection d'intrusion. N'ayant pu tester la détection d'intrusion en situation réelle, nous avons choisi de mener nos évaluations sur des enregistrements de trafic CAN que nous modifions afin d'intégrer les scénarios d'attaques que nous cherchons à détecter. Ne possédant pas de nombreux exemples de scénarios d'attaques contre des réseaux embarqués, a fortiori transposables au véhicule sur lequel nous avons pris nos enregistrements, nous avons dû utiliser une même source (les spécifications du système) pour générer les attaques et les mécanismes de défense. Il nous est donc malaisé de déterminer avec précision la couverture de notre système. Cependant, nous pouvons noter que les mécanismes implémentant des tests de cohérence n'ont comme prévu pas levé de faux positifs lors de nos analyses.

Concernant les performances de notre système, là aussi, les résultats de nos expériences correspondent à la théorie présentée dans le chapitre précédent. La méthode utilisant un AFD $A_{Attacks}$ effectue bien ses analyses en temps constant alors que la méthode utilisant A_{right} utilise un automate plus petit mais nécessite plus de temps pour analyser une trame en moyenne. Sur des systèmes de petite et moyenne taille, les performances obtenues avec les deux méthodes sont satisfaisantes et permettent d'analyser l'intégralité du trafic CAN d'une automobile en temps réel. En revanche,

pour des systèmes de très grande taille, pour lesquels il deviendrait trop coûteux de stocker un automate $A_{attacks}$, il est possible que le temps nécessaire pour analyser une trame avec A_{right} devienne trop grand (de l'ordre de la milliseconde), ce qui risque d'entraîner des retards dans l'analyse. Cependant, comme nous l'avons vu dans le chapitre précédent, l'efficacité de cette méthode augmente avec le nombre de trames déjà analysées puisque la taille de S_{right} diminue exponentiellement avec la durée de l'observation.

Par ailleurs, nous ne nous sommes concentrés ici que sur l'évaluation des différents mécanismes de détection, particulièrement les tests de cohérence qui constituent notre principale contribution. Cependant, comme nous l'avons déjà mentionné, il n'est peut être pas nécessaire d'alerter le conducteur à la première anomalie détectée, surtout si celle-ci concerne un système peu critique. La conception d'un mécanisme qui corrèle les différentes alertes levées par notre système afin de prendre une décision adaptée doit être la prochaine étape dans la conception d'un IDS véritablement autonome. Nous avons mentionné à ce sujet les travaux présentés dans [MGF10] qui nous semblent constituer une piste de réflexion intéressante.

Conclusion générale

L'augmentation constante du nombre de fonctionnalités dévolues aux calculateurs embarqués et les multiples capacités de communication dont sont désormais équipées les automobiles ont fait émerger de nombreuses problématiques de sécurité-immunité auxquelles les constructeurs se doivent de répondre. Afin de protéger au mieux le système complexe que représente une voiture, la mise en place des politiques de sécurité doit se faire en ayant une vision d'ensemble des systèmes concernés et de la diversité des attaques qui peuvent les cibler. Ainsi, le déploiement de mécanismes de sécurité préventifs, visant à empêcher un attaquant de s'introduire dans le système, est une solution nécessaire mais pas suffisante. En effet, un moyen de protection, même très bien conçu, peut parfois être contourné, par exemple grâce à de nouvelles techniques encore inconnues au moment de sa conception. Ainsi, la mise en place de mécanismes dont le but est de détecter la présence d'un attaquant au sein même du système permet d'établir une ligne de défense supplémentaire lorsqu'une intrusion a réussi. Au vu de la grande diversité d'architectures réseaux embarquées dans les véhicules actuels, des contraintes spécifiques à la conception de systèmes embarqués dans l'automobile et du caractère relativement récent des problématiques de sécurité-immunité liées aux systèmes embarqués dans les automobiles, l'objectif principal de cette thèse était de proposer une première implémentation d'un système de détection d'intrusions adapté à des architectures de réseaux embarqués actuelles (utilisant CAN) et dont les principes pourront être adaptés aux véhicules de demain.

Dans ce manuscrit, nous avons tout d'abord présenté les caractéristiques des systèmes embarqués dans l'automobile, en nous intéressant particulièrement aux protocoles de communication employés sur leurs réseaux et plus spécifiquement au plus employé d'entre eux (et standard *de facto*) à l'heure actuelle : CAN. Nous avons ensuite étudié les vulnérabilités de ces réseaux ainsi que les différents scénarios d'attaques possible contre un véhicule moderne, pour lesquels nous avons proposé une classification. Après avoir établi un état de l'art des différents mécanismes de sécurité pouvant être appliqués aux systèmes embarqués dans une automobile, nous avons présenté le fonctionnement de notre système de détection d'intrusions. Celui-ci fait actuellement l'objet d'une demande de brevet [1]. Notre principale contribution dans ce domaine est la description d'une méthode permettant d'effectuer une corrélation entre les trames observées sur le réseau :

1. À partir de spécifications décrivant le fonctionnement des systèmes embarqués sous la forme de machines à état, nous générons un langage régulier décrivant un ensemble de séquences interdites.
2. Ce langage est représenté par un automate fini déterministe qui est alors parcouru lors de l'analyse en ligne des trames circulant sur le réseau.

1. demande FR1459640 déposée le 8 octobre 2014

3. Dans l'hypothèse d'une explosion du nombre d'états nécessaires pour représenter un automate fini déterministe décrivant le langage d'attaques, nous avons proposé une méthode alternative de détection utilisant un automate de taille bornée mais possédant une plus forte complexité temporelle

Enfin, des premières expérimentations menées sur des cas d'utilisation relativement simples nous ont permis de démontrer la faisabilité de nos méthodes. Ces expérimentations nous ont également permis d'exprimer les avantages et les limites de notre approche. Ces analyses doivent être confirmées par des essais à l'échelle d'un véhicule complet.

Au delà de la nécessité d'effectuer des essais à plus grande échelle avant d'envisager un déploiement au sein d'un véhicule pouvant être commercialisé, la conduite de ces travaux nous permet d'envisager d'autres perspectives de recherche.

Tout d'abord, suite aux différents scénarios d'attaques et aux vulnérabilités identifiées dans le chapitre 2, la mise en place de recommandations et de standards concernant la sécurité-immunité des systèmes embarqués dans les automobiles nous semble nécessaire, similairement à ce qui a été fait concernant la sécurité-innocuité de ces systèmes (ISO 26262). Une coopération entre constructeurs et experts en sécurité, comme ce fut le cas pour les projets de sécurisation des communications Car2X présentés dans le chapitre 3, permettrait ainsi d'établir de tels standards (et d'éventuelles certifications associées), afin de concevoir des véhicules plus sûrs en limitant les surcoûts et de rassurer une clientèle de plus en plus préoccupée par les questions de sécurité informatique.

Par ailleurs, lors de nos travaux, nous nous sommes concentrés exclusivement sur le protocole CAN car les systèmes sur lesquels nous pouvions expérimenter communiquaient via des bus CAN. Or, comme nous l'avons vu dans le chapitre 1, il existe de nombreux autres protocoles de communication utilisables dans les automobiles et CAN est amené à perdre son statut de standard *de facto* à plus ou moins long terme. Ainsi, si dans le cas de CAN, nous pouvons nous permettre d'analyser en temps réel et en profondeur toutes les trames circulant sur le réseau, l'utilisation de protocoles présentant des débits plus élevés et transportant des données plus complexes pourrait remettre en question ce principe. Dans les cas où de telles technologies sont employées, il convient alors de vérifier la faisabilité de notre méthode, ou du moins d'adapter celle-ci pour tenir compte des spécificités de ces réseaux. De même, une éventuelle mise en place de mécanismes de chiffrement ou de signature des messages sur les réseaux internes impliquera aussi d'adapter notre IDS pour la prendre en compte. Cependant, au delà du temps traitement supplémentaire correspondant au déchiffrement des messages ou à la vérification d'une signature (dans les cas où cela s'avérerait nécessaire), les principes que nous avons employés pour la détection d'intrusion en tant que telle (règles de contrôle et utilisation d'automates finis) ne devraient pas être impactés.

De plus, les architectures réseaux des véhicules évoluent avec chaque nouveau modèle. La conception d'une version distribuée de notre système de détection d'intrusion, consistant en différents modules répartis sur les différents domaines du

réseau, permettrait de bénéficier d'une grande diversité de sources de données au sein d'architectures toujours plus complexes.

Enfin, dans l'optique du développement de mécanismes de sécurité réactifs, la détection d'intrusion ne constitue que la première étape. Or, dans nos travaux, nous nous sommes exclusivement concentrés sur la détection de potentielles intrusions dans un réseau automobile embarqué. La solution que nous avons présentée dans ce manuscrit consiste ainsi en un système passif. La problématique suivante devient alors de déterminer comment réagir en fonction de la menace détectée. Différentes possibilités peuvent être envisagées : 1) se contenter d'enregistrer la séquence détectée dans le cadre d'éventuelles analyses *forensics*, 2) concevoir des mécanismes de réaction afin de limiter l'impact de l'attaque tout en respectant les principes régissant la *safety* du véhicule ou encore 3) alerter le conducteur.

- Concernant l'utilisation des enregistrements, un système relayant régulièrement les alertes générées vers des serveurs distants permettrait d'obtenir un aperçu global des incidents de sécurité relatifs à une flotte de véhicules. Cependant, la mise en place d'un tel système soulève des problématiques de gestion des données et de respect de la vie privée qu'il conviendra de traiter.
- La conception d'un système de prévention d'intrusions (IPS) constitue logiquement la principale évolution envisagée pour notre système. Les possibilités de réaction à une attaque donnée sont nombreuses et chacune constitue une problématique intéressante. Parmi les possibilités envisageables, nous pouvons citer la mise en place de mécanismes de transmission des alertes sur le réseau ou la conception d'un système permettant de bloquer une trame suspecte en cours d'émission ou d'ignorer les futures émissions d'un calculateur jugé corrompu (certains exemples de la littérature proposent ainsi de saturer le bus en réponse à une attaque [MV14], [MHT+12], indifféremment de la gravité de l'attaque, ce qui entraîne le passage de tous les ECU en mode dégradé à chaque anomalie détectée). Cependant, pour chacun de ces cas, comme le système n'est plus passif, il conviendra de s'assurer que les fonctionnalités ainsi implémentées ne puissent pas à leur tour être détournées par un attaquant.
- Plus généralement, la détermination du type de réaction à adopter en fonction de la menace détectée doit faire l'objet d'études plus approfondies. Ainsi, concernant le signalement d'alertes de sécurité au conducteur, il n'est pas envisageable de détourner son attention à la moindre anomalie détectée, mais seulement en cas d'urgence. Une solution possible, proposée dans [HKD08], consiste en plusieurs moyens de signalisation (visuels, sonores ou haptiques) en fonction de la gravité de l'incident détecté.

Bibliographie

[AFS97] William A Arbaugh, David J Farber et Jonathan M Smith : A secure and reliable bootstrap architecture. *In Proc. Symp. Security and Privacy*, pages 65–71, Oakland, CA, 1997. IEEE. (Cité en page 58.)

[ALRL04] Algirdas Avizienis, J-C Laprie, Brian Randell et Carl Landwehr : Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, 2004. (Cité en page 19.)

[Ari12] Feasible Car Cyber Defense. *In 10th Embedded Security in Cars Conference (ESCAR)*. Arilou Technologies, 2012. (Cité en page 62.)

[BCM+03] Marie-Pierre Béal, Maxime Crochemore, Filippo Mignosi, Antonio Restivo et Marinella Sciortino : Computing forbidden words of regular languages. *Fundamenta Informaticae*, 56(1-2):121–136, 2003. (Cité en page 96.)

[Bel11] Lucia Lo Bello : The case for ethernet in automotive communications. *SIGBED Rev.*, 8(4):7–15, décembre 2011. (Cité en page 17.)

[BJZ10] Janusz Brzozowski, Galina Jirásková et Chenglong Zou : Quotient complexity of closed languages. *In Computer Science–Theory and Applications*, pages 84–95. Springer, 2010. (Cité en page 91.)

[Bro05] Manfred Broy : Automotive software and systems engineering. *In Formal Methods and Models for Co-Design, 2005. MEMOCODE'05. Proceedings. Third ACM and IEEE International Conference on*, pages 143–149. IEEE, 2005. (Cité en page 5.)

[BSDT09] RR Brooks, S Sander, Juan Deng et Joachim Taiber : Automobile security concerns. *IEEE Veh. Technol. Mag.*, 4(2):52–64, 2009. (Cité en page 34.)

[BSWE13] Alexandre Bouard, Hendrik Schweppe, Benjamin Weyl et Claudia Eckert : Leveraging in-car security by combining information flow monitoring techniques. *In VEHICULAR 2013, The Second International Conference on Advances in Vehicular Systems, Technologies and Applications*, pages 1–7, 2013. (Cité en page 60.)

[CAN12] CAN with Flexible Data-Rate - Specification version 1.0. `http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can_fd_spec.pdf`, 2012. [Online ; accessed February-2015]. (Cité en page 56.)

[CBW08] Nicolas Courtois, Gregory Bard et David Wagner : Algebraic and slide attacks on keeloq. *In Fast Software Encryption*, pages 97–115, Lausanne, Switzerland, 2008. Springer. (Cité en page 43.)

[Cha09] Robert N Charette : This car runs on code. *IEEE Spectr.*, 46(3):3, 2009. (Cité en pages 5 et 7.)

[CMK+11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno *et al.* : Comprehensive experimental analyses of automotive attack surfaces. *In Proc. 20th USENIX Security*, San Francisco, CA, 2011. (Cité en pages 39, 40, 41, 42, 44, 45 et 76.)

[DBF91] Yves Deswarte, Laurent Blain et J-C Fabre : Intrusion tolerance in distributed computing systems. *In Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 110–121. IEEE, 1991. (Cité en page 22.)

[DVD11] Leandro D'Orazio, Filippo Visintainer et Marco Darin : Sensor networks on the car : State of the art and future challenges. *In Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, Grenoble, France, 2011. (Cité en page 17.)

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh et Mohammad Shalmani : On the power of power analysis in the real world : A complete break of the keeloq code hopping scheme. *Advances in Cryptology*, pages 203–220, 2008. (Cité en page 43.)

[EM13] Christopher E. Everett et Damon McCoy : Octane (open car testbed and network experiments) : Bringing cyber-physical security research to researchers and students. *In Presented as part of the 6th Workshop on Cyber Security Experimentation and Test*, Washington, D.C., 2013. USENIX. (Cité en page 37.)

[FDC10] Aurélien Francillon, Boris Danev et Srdjan Capkun : Relay attacks on passive keyless entry and start systems in modern cars. *IACR ePrint Report*, 2010/332, 2010. (Cité en page 43.)

[GHR+09] André Groll, Jan Holle, Christoph Ruland, Marko Wolf, Thomas Wollinger et Frank Zweers : Oversee a secure and open communication and runtime platform for innovative automotive applications. *In 7th Embedded Security in Cars Conference (ESCAR)*, Düsseldorf, Germany, 2009. (Cité en pages 58 et 59.)

[GMVHV12] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege et Ingrid Verbauwhede : Libra-can : a lightweight broadcast authentication protocol for controller area networks. *In Proc. 11th Int. Conference Cryptology and Network Security, CANS*, Darmstadt, Germany, 2012. (Cité en page 56.)

[HB95] L. R. Halme et K. R. Bauer : AINT misbehaving : A taxonomy of anti-intrusion techniques. *Computers and Security*, 14:606–606, 1995. (Cité en pages 52 et 53.)

[HD07] Tobias Hoppe et Jana Dittman : Sniffing/replay attacks on can buses : A simulated attack on the electric window lift classified using an adapted cert taxonomy. *In Proc. 2nd Workshop on Embedded Systems Security (WESS)*, Salzburg, Austria, 2007. (Cité en page 34.)

[HKD08] Tobias Hoppe, Stefan Kiltz et Jana Dittmann : Adaptive dynamic reaction to automotive it security incidents using multimedia car environment. *In 4th Int. Conference on Information Assurance and Security*, pages 295–298. IEEE, 2008. (Cité en pages 66 et 125.)

[HKD09] Tobias Hoppe, Stefan Kiltz et Jana Dittmann : Automotive it-security as a challenge : Basic attacks from the black box perspective on the example of privacy threats. *Computer Safety, Reliability, and Security*, pages 145–158, 2009. (Cité en pages 32, 35, 37, 62 et 76.)

[HKD11] Tobias Hoppe, Stefan Kiltz et Jana Dittmann : Security threats to automotive can networks—practical examples and selected short-term countermeasures. *Reliability Engineering & System Safety*, 96(1):11–25, 2011. (Cité en pages 56, 71 et 73.)

[HL98] John D Howard et Thomas A Longstaff : A common language for computer security incidents. Tech. Rep. SAND98-8667, Sandia National Laboratories, 1998. (Cité en page 34.)

[HRS+09] Olaf Henniger, Alastair Ruddle, Hervé Seudié, Benjamin Weyl, Marko Wolf et Thomas Wollinger : Securing vehicular on-board it systems : The EVITA project. *In 25th VDI/VW Automotive Security Conference*, Ingolstadt, Germany, 2009. (Cité en pages 13 et 56.)

[ISO11] ISO 26262 - road vehicles, functional safety. `http://www.iso.org/iso/catalogue_detail?csnumber=43464`, novembre 2011. (Cité en pages 1 et 23.)

[KCR+10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson et Hovav Shacham : Experimental security analysis of a modern automobile. *In IEEE Symp. Security and Privacy*, pages 447–462, Oakland, CA, 2010. (Cité en pages 37, 38, 39, 76 et 119.)

[LAD+96] Jean-Claude Laprie, Jean Arlat, Yves Deswarte, David Powell, Karama Kanoun, Mohamed Kaâniche et Yves Crouzet : *Guide de la sûreté de fonctionnement*. Cépaduès, 1996. (Cité en page 19.)

[LKS05] Aleksandar Lazarevic, Vipin Kumar et Jaideep Srivastava : Intrusion detection : A survey. *In* Vipin Kumar, Jaideep Srivastava et Aleksandar Lazarevic, éditeurs : *Managing Cyber Threats*, volume 5 de *Massive Computing*, pages 19–78. Springer US, 2005. (Cité en page 64.)

[LNJ08] Ulf E Larson, Dennis K Nilsson et Erland Jonsson : An approach to specification-based attack detection for in-vehicle networks. *In*

Intelligent Vehicles Symposium, pages 220–225, Eindhoven, Netherlands, 2008. IEEE. (Cité en pages 66, 67, 78 et 83.)

[MA11] Michael Müter et Naim Asaj : Entropy-based anomaly detection for in-vehicle networks. *In Intelligent Vehicles Symposium (IV)*, pages 1110–1115, Baden Baden, Germany, 2011. IEEE. (Cité en pages 67 et 79.)

[MAF03] MAFTIA, Malicious and Accidental Fault Tolerance in Internet Applications : conceptual model and architecture. Maftia project deliverable d21, Project IST-1999-11583, 2003. (Cité en page 22.)

[Mar15] Tracking & hacking : Security & privacy gaps put american drivers at risk. `http://www.markey.senate.gov/imo/media/doc/2015-02-06_MarkeyReport-Tracking_Hacking_CarSecurity2.pdf`, février 2015. (Cité en page 25.)

[MFRV14] Hélène Martorell, Jean-Charles Fabre, Matthieu Roy et Régis Valentin : Improving adaptiveness of autosar embedded applications. *In Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 384–390. ACM, 2014. (Cité en page 8.)

[MG14] Pal-Stefan Murvay et Bogdan Groza : Source identification using signal characteristics in controller area networks. *IEEE Signal Process. Lett.*, 21(4):395–399, 2014. (Cité en page 68.)

[MGF10] Michael Müter, André Groll et Felix C Freiling : A structured approach to anomaly detection for in-vehicle networks. *In 6th Int. Conference Information Assurance and Security (IAS)*, pages 92–98, Atlanta, GA, 2010. IEEE. (Cité en pages 66, 67, 79, 84 et 121.)

[MHT+12] Tsutomu Matsumoto, Masato Hata, Masato Tanabe, Katsunari Yoshioka et Kazuomi Oishi : A method of preventing unauthorized data transmission in controller area network. *In Vehicular Technology Conference (VTC Spring)*, pages 1–5, Yokohama, Japan, 2012. IEEE. (Cité en pages 67, 78 et 125.)

[Mic03] Cedric Michel : *Langage de description d'attaques pour la détection d'intrusions par corrélation d'évènements ou d'alertes en environnement réseau hétérogène.* Thèse de doctorat, Université de Rennes 1, 2003. (Cité en pages 64 et 65.)

[MLLS12] Rim Moalla, Houda Labiod, Brigitte Lonc et Noémie Simoni : Risk analysis study of ITS communication architecture. *In 3rd Int. Conference Network of the Future*, pages 1–5, Tunis, Tunisia, 2012. (Cité en page 43.)

[MLLS13] Rim Moalla, Brigitte Lonc, Houda Labiod et Noémie Simoni : Security architecture for cooperative its-s vehicles. *In Escar 2013*, 2013. (Cité en page 54.)

[Mur10] C. Murray : Automakers opting for model-based design. *Design News*, mai 2010. (Cité en page 8.)

[MV13] Charlie Miller et Chris Valasek : Adventures in automotive networks and control units. *Last Accessed from h ttp ://illmatics. com/-car_ hacking. pdf on*, 2013. (Cité en pages 37, 38, 76 et 116.)

[MV14] Charlie Miller et Chris Valasek : A survey of remote automotive attack surfaces. *Last Accessed from http ://illmatics.com/remote attack surfaces.pdf*, 2014. (Cité en pages 5, 11, 12, 61, 63, 67, 79, 80 et 125.)

[NL08] Dennis K Nilsson et Ulf E Larson : Simulated attacks on can buses : vehicle virus. *In Proc. 5th IASTED Int. Conference on Communication Systems and Networks*, pages 66–72, Langkawi, Malaysia, 2008. (Cité en page 38.)

[NL09] Dennis K Nilsson et Ulf E Larson : A defense-in-depth approach to securing the wireless vehicle infrastructure. *Journal of Networks*, 4(7):552–564, 2009. (Cité en page 53.)

[NLPJ09] Dennis K Nilsson, Ulf E Larson, Francesco Picasso et Erland Jonsson : A first simulation of attacks in the automotive network communications protocol flexray. *In Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08*, pages 84–91. Springer, 2009. (Cité en page 33.)

[NPL08] Dennis K Nilsson, Phu H Phung et Ulf E Larson : Vehicle ecu classification based on safety-security characteristics. *In Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference, IET*, pages 1–7. IET, 2008. (Cité en page 24.)

[NSA12] Defense in depth - a practical strategy for achieving Information Assurance in today's highly networked environments. `https://www.nsa.gov/ia/_files/support/defenseindepth.pdf`, 2012. [Online ; accessed February-2015]. (Cité en page 51.)

[NSL09] Nicolas Navet et Françoise Simonot-Lion : Trends in automotive communication systems. *Embedded Systems Handbook : Networked Embedded Systems-2nd ed.*, pages 13–1, 2009. (Cité en page 10.)

[RMM+10] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe et Ivan Seskar : Security and privacy vulnerabilities of in-car wireless networks : A tire pressure monitoring system case study. *In Proc. USENIX Security Symposium*, pages 323–338, Washington, DC, 2010. (Cité en pages 25 et 43.)

[SAN+15] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaâniche et Youssef Laarouchi : A language-based intrusion detection approach for automotive embedded networks. *In 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, Zhangjiajie, China, 2015. (Cité en page 27.)

[SEJ08] Randy Smith, Cristian Estan et Somesh Jha : Xfa : Faster signature matching with extended automata. *In IEEE Symposium on Security and Privacy*, pages 187–201. IEEE, 2008. (Cité en page 120.)

[SNA+13a] Ivan Studnia, Vincent Nicomette, Éric Alata, Yves Deswarte, Mohamed Kaâniche et Youssef Laarouchi : Security of embedded automotive networks : state of the art and a research proposal. *In SAFECOMP 2013-Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013. (Cité en page 27.)

[SNA+13b] Ivan Studnia, Vincent Nicomette, Eric Alata, Yves Deswarte, Mohamed Kaâniche et Youssef Laarouchi : Survey on security threats and protection mechanisms in embedded automotive networks. *In 2nd Workshop on Open Resilient Human-aware Cyber-Physical Systems*, Budapest, Hungary, 2013. (Cité en page 27.)

[SNAK12] Ivan Studnia, Vincent Nicomette, Éric Alata et Mohamed Kaâniche : Plateforme distribuée de pots de miel haute-interaction et résultats expérimentaux. *In 7eme Conférence sur la Sécurité des Architectures Réseaux et Systèmes d'Information (SARSSI)*, 2012. (Cité en page 68.)

[SP08] Karen Scarfone et John Padgette : Guide to bluetooth security. *NIST Special Publication*, 800:121, 2008. (Cité en page 49.)

[SPS+09] Sandro Schulze, Mario Pukall, Gunter Saake, Tobias Hoppe et Jana Dittmann : On the need of data management in automotive systems. *In BTW*, volume 144, pages 217–226, 2009. (Cité en page 60.)

[SR12] Hendrik Schweppe et Yves Roudier : Security and privacy for in-vehicle networks. *In Vehicular Communications, Sensing, and Computing (VCSC)*, pages 12–17, Seoul, Korea, 2012. IEEE. (Cité en page 60.)

[SRW+11] Hendrik Schweppe, Yves Roudier, Benjamin Weyl, Ludovic Apvrille et Dirk Scheuermann : Car2x communication : securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography. *In Vehicular Technology Conference (VTC Fall)*, pages 1–5, San Francisco, CA, 2011. IEEE. (Cité en page 57.)

[TCG11] TPM main specification. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, 2011. [Online ; accessed February-2013]. (Cité en page 56.)

[VHSV11] Anthony Van Herrewege, Dave Singelee et Ingrid Verbauwhede : Canauth-a simple, backward compatible broadcast authenti-

cation protocol for can bus. *In 9th Embedded Security in Cars Conference*, Dresden, Germany, 2011. (Cité en page 56.)

[VNLJ08] Vilhelm Verendel, Dennis K Nilsson, Ulf E Larson et Erland Jonsson : An approach to using honeypots in in-vehicle networks. *In 68th Vehicular Technology Conference*, pages 1–5, Calgary, Canada, 2008. IEEE. (Cité en pages 35, 68 et 70.)

[WWP04] Marko Wolf, André Weimerskirch et Christof Paar : Security in automotive bus systems. *In 2nd Embedded Security in Cars Workshop (ESCAR 2004)*, pages 11–12, Bochum, Germany, 2004. (Cité en page 32.)

[WWW07] Marko Wolf, André Weimerskirch et Thomas Wollinger : State of the art : Embedding security in vehicles. *EURASIP Journal on Embedded Systems*, 2007. (Cité en page 34.)

[Yu01] S Yu : State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001. (Cité en page 91.)

[ZWT09] Tobias Ziermann, Stefan Wildermann et Jürgen Teich : Can+ : A new backward-compatible controller area network (can) protocol with up to 16× higher data rates. *In Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.*, pages 1088–1093. IEEE, 2009. (Cité en page 56.)

Résumé

Les calculateurs embarqués dans les automobiles, ou ECU (*Electronic Control Unit*) sont responsables d'un nombre croissant de fonctionnalités au sein du véhicule. Pour pouvoir coordonner leurs actions, ces calculateurs s'échangent des données via des bus de communication et forment ainsi un véritable réseau embarqué. Si historiquement ce réseau pouvait être considéré comme un système fermé, l'apparition de nombreux moyens de communication dans les automobiles a ouvert ce réseau au monde extérieur et fait émerger de nombreuses problématiques de sécurité dans ce domaine.

Nos travaux s'inscrivent dans une démarche de mise en place de moyens de sécurité-immunité dans les réseaux automobiles. La thématique de la sécurité-immunité dans l'automobile étant un sujet relativement récent, un effort particulier a été apporté à la définition du contexte. Ainsi, dans ce manuscrit, nous décrivons les menaces qui peuvent cibler ces systèmes embarqués, proposons une classification des scénarios d'attaques puis présentons les différents mécanismes de sécurité pouvant être appliqués aux systèmes embarqués d'une automobile.

Ensuite, afin de compléter les mesures de sécurité préventives mises en place pour empêcher un attaquant de pénétrer au cœur du réseau embarqué, nous proposons dans cette thèse un système de détection d'intrusion pour les réseaux automobiles embarqués. Celui-ci, conçu à partir des spécifications du ou des systèmes à surveiller, intègre notamment des mécanismes permettant d'effectuer une corrélation des messages observés sur le réseau afin d'identifier des séquences de messages suspectes. Après avoir décrit formellement le fonctionnement de notre système de détection, nous présentons de premières expérimentations visant à valider notre méthode et à évaluer ses performances.

Mots clés : Sécurité, Automobile, Détection d'Intrusions, Réseaux Embarqués

---

Abstract

In today's automobiles, embedded computers, or ECUs (Electronic Control Units) are responsible for an increasing number of features in a vehicle. In order to coordinate their actions, these computers are able to exchange data over communication buses, effectively constituting an embedded network. While this network could previously be considered a closed system, the addition of means of communication in automobiles has opened this network to the outside world, thus raising many security issues.

Our research work focuses on these issues and aims at proposing efficient architectural security mechanisms for protecting embedded automotive networks. The security of embedded automotive systems being a relatively recent topic, we first put a strong focus on defining the context. For that purpose, we describe the threats that can target a car's embedded systems, provide a classification of the possible attack scenarios and present a survey of protection mechanisms in embedded automotive networks.

Then, in order to complement the preventive security means that aim at stopping an attacker from entering the embedded network, we introduce an Intrusion Detection System (IDS) fit for vehicular networks. Leveraging the high predictability of embedded automotive systems, we use language theory to elaborate a set of attack signatures derived from behavioral models of the automotive calculators in order to detect a malicious sequence of messages transiting through the internal network. After a formal description of our IDS, we present a first batch of experiments aimed at validating our approach and assessing its performances.

Keywords : Security, Automotive, Intrusion Detection, Embedded Networks

---