Building an automotive security assurance case using systematic security evaluations

Cheah, M, Shaikh, S, Bryans, J & Wooderson, P

DOI 10.1016/j.cose.2018.04.008
ISSN 0167-4048

Publisher: Elsevier

Building an automotive security assurance case using systematic security evaluations

Madeline Cheah[a], Siraj A. Shaikh[b], Jeremy Bryans[b], Paul Wooderson[a]

[a]*HORIBA MIRA, Nuneaton, CV10 0TU, United Kingdom*
[b]*Research Institute for Future Transport and Cities, Coventry University, CV1 5FB, Coventry, United Kingdom*

Abstract

Security testing and assurance in the automotive domain is challenging. This is predominantly due to the increase in the amount of software and the number of connective entry points in the modern vehicle. In this paper we build on earlier work by using a systematic security evaluation to enumerate undesirable behaviours, enabling the assignment of severity ratings in a (semi-) automated manner. We demonstrate this in two case studies; firstly with the native Bluetooth connection in an automotive head unit, and secondly with an aftermarket diagnostics device. We envisage that the resulting severity classifications would add weight to a security assurance case, both as evidence and as guidance for future test cases.

Keywords: automotive, Bluetooth, cybersecurity, security assurance, penetration testing

1. Introduction

Historically, embedded systems were designed to operate in tightly-controlled environments which required specialist knowledge to design, calibrate and deploy. Developments in functionality and connectivity, however, have meant that the amount of software and its concomitant complexity has increased dramatically.

There are several trends which have contributed to the automotive threat landscape, each of which lead to increased attack surface area and increased complexity, which impairs testability:

- Firstly, the presence of an increased volume of software (measured, for example, in lines of code) to meet the requirements of an increasingly sophisticated functionality and attendant rise in the number of processing units. This leads to compounded complexity. There are more lines of code within some of the more advanced luxury vehicles than a fighter jet [1]. Subsequently, testability and auditing (and in this context, security testing) becomes more difficult and the likelihood of large numbers and the severity of vulnerabilities increases [2].

- Secondly, there has been significant development and integration of (wireless) communication interfaces, which means more connectivity. Subsequently, this has led to increased number of connections in the intra-vehicular network as reuse of externally provided information becomes more important. There is also a concomitant rise in the number of external peripheral devices that can now connect to the vehicle. This means that there are now more access points for malicious attackers, and also potentially negatively impacts system boundaries by blurring them, or extending them to beyond the control of original manufacturers such that unknown interactions (and therefore possible security risks) could exist.

- Finally content volume, variability and value has changed and increased, which means that there is more data about the vehicle to extract. Additionally, the data that is extracted is potentially more valuable if it is personal data that is obtained. In summary, there is more of and more kinds of data to consider and defend.

Security engineering (and security testing as part of that process) is still relatively novel in mainstream automotive production [3, 4], and typically security is incidental and a by-product of achieving performance and safety goals [5, 6, 7, 8]. Even with advanced formal methods for modelling and testing, the need for and number of demands, features and increased connecting power means that, even had security been considered, the scale of the problem facing security testers is now much broader [9].

There are challenges to introducing a security engineering process into the automotive industry. Vehicles are heterogeneous with many variants and configurations. Production costs are based on units, with narrow profit margins. There are differences between life and development cycles to traditional computing software and hardware. Supply chains adds to the level of obscurity with each tier with regards to the final system [2].

More specifically, securing interfaces in vehicles also comes with many issues. Any security mechanism will require additional processing overhead, and on the hardware level, has ramifications in provision of energy and in physical assembly and design, such as placement of additional wiring.

Even should such concerns be addressed, countermeasures that are commonly used currently for large and complex systems are not suitable for vehicular embedded systems because of hardware constraints and the differences in network configuration. Well-established defences at software level such as the use of cryptography, firewalls and intrusion detection systems (IDS) cannot be implemented without considerable change in architecture due to the use of sufficiently different protocols and topologies within the automotive domain, and the potentially costly computational power required. Post-release, maintenance becomes an issue as patches for discovered vulnerabilities, unless performed over-the-air, are difficult to apply once units are sold.

Although the vast majority of demonstrated attacks that have been directed at the vehicle use automotive-specific vectors, many of the methods are familiar to security professionals. This includes the use of malware and known software vulnerabilities, proximity extending hardware, replay attacks or simply reverse engineering messages to gain illicit knowledge of the system [8]. Considering the similarity of attack methods, parallels can be drawn between non-automotive and automotive systems, forming a baseline from which to draw information on possible weaknesses.

All of the above is dependent on acquiring knowledge and information regarding existing vulnerabilities. From the number and variety of reported threats, weaknesses and exploits (see Section 2), all of which have been demonstrated as feasible, it is clear that a methodical description of the problem, as well as a systematic method of establishing relative priority regarding risks and threats, is required. Furthermore, manual methods, whilst useful, can be time-consuming. This is similar to the problem facing those who are addressing vehicle safety; however, the process of evaluation in vehicle safety is well-established. We endeavour to address the problem in vehicle security in a similar fashion, using industry standards and to aid in the automation of the process.

The main contribution of this paper is the semi-automated classification of results using security severity ratings (in line with industry standards), resulting from a systematic security evaluation of automotive Bluetooth interfaces. We present two case studies, one which evaluates a native automotive Bluetooth interface, and one which evaluates a aftermarket Bluetooth-enabled device attached to the vehicle's on-board diagnostics port.

The evaluation is from a black box perspective. This is because, although there are high-confidence formal methods for security testing (see Section 2), the primary barrier to using such methods is that the information required to do so is not available, both due to commercial confidentiality and the obscurity of sub-components within the system (many of which are third party)[10]. This also precludes other methods of enabling systematic evaluation such as attack graphs, for which formal model checking could be performed.

Thus, we use a black box approach to test the case study systems (see experimental applications in Section 4), and we build here on our earlier work [11], by including severity classifications as part of the overall systematic security evaluation. These classifications can inform the selection of future test cases, and can act as evidence within a security assurance case.

The rest of the paper is structured as follows:

- We explore related work, including the exposure of vulnerabilities in vehicles, comparative methods and severity classifications in Section 2;

- We present our methodology together with the fundamental concepts that underlie it in Section 3;
- We briefly outline the implementation of our methodology as a proof-of-concept tool in Section 3.4;
- We present the results of two experiments, one in which we apply our experimental application to the Bluetooth in an infotainment unit and one where we study an aftermarket on-board diagnostic device) in Section 4;
- We look at how the results could be used in a security assurance case in Section 5 and finally,
- We discuss implications, give concluding remarks and explore future directions of research in Section 6.

2. Related Work

In this section, we discuss security testing in the automotive domain, with a brief overview of vulnerabilities exposed and the necessity for *systematic* security testing (Section 2.1). We then explore comparative methods and schemes that allow for a security classification or assurance with the results of such testing in Section 2.2.

2.1. Security Testing

Many initial studies looking at automotive security have been exploratory, with demonstrated attacks on the vehicle as a whole [8, 12], on components or sub-components [13], on the intra-vehicular network [14, 15] or on peripheral devices that connect to the vehicle such as through the On-Board Diagnostics (OBD-II) port [16, 17].

These papers established the nature of vulnerabilities and are impressive in their depth of experimentation. They make clear the need for a systematic testing method that would help highlight the problems an engineer must address.

A systematic security evaluation method has many advantages. There is a disparity between what an attacker must find in order to exploit the system (potentially just one vulnerability) and the number of flaws a defender would have to safeguard in order to protect the system (as many as possible). An ad-hoc approach to finding vulnerabilities could mean that bugs and weaknesses are overlooked [18]. A methodical approach increases the likelihood of determining the nature and number of problems [19]. The latter is essential since information sharing is still limited due to the competitive nature of the industry [20]. Systematic analyses can also be supported by a variety of tools and utilities (as is the case in this paper) and be further expanded upon once more information regarding the system is known.

Comparative methods include model-based security testing [19], model-driven engineering [21] or using "anti-models" [22] or abuse cases [23] to achieve systematism. What these all have in common, however, is the need to have pre-built models (or enough technical information to generate a trustworthy model) in order to run or generate any test cases.

2.2. Automotive Specific Severity Classifications

The "E-Safety Vehicle Intrusion Protected Applications" (EVITA) project [24] ultimately aims to provide a secure architecture for automotive on-board networks and evaluates the realisation of this using two "views". The first of these is the *magnified view*. Of especial interest within this are the automotive specific systematic methods of evaluation described. Attack tree modelling (discussed further in Section 3.2) is used to support these processes. Of particular interest is the classification of the severity of various outcomes (Table 1).

Severity levels have also featured in other automotive security projects, such as the "Healing vulnerabilities to enhance software security and safety" (HEAVENS) project [25]. This project is aimed at facilitating security requirements engineering. It uses the popular threat modelling method STRIDE (a mnemonic for Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege) [26] for threat analysis, ending with the assignment of

Table 1: EVITA Severity Classification for Automotive Security Threats

Severity Classes	*Classes of harm to stakeholders*			
	Safety (S_s)	*Privacy (S_p)*	*Financial (S_f)*	*Operational (S_o)*
0	No injuries	No unauthorised access to data	No financial loss	No impact on operational performance
1	Light or moderate injuries	Anonymous data only	Low-level financial loss	Operational impact not discernible to driver
2	Severe and life-threatening injuries (survival probable) or light/moderate injuries for multiple vehicles	Identification of vehicle or driver	Moderate financial loss, or low losses for multiple vehicles	Driver aware of performance degradation, or indiscernible operational impacts for multiple vehicles
3	Life threatening (survival uncertain) or fatal injuries, or severe injuries for multiple vehicles	Driver or vehicle tracking, or identification of driver or vehicle for multiple vehicles	Heavy financial loss, or moderate losses for multiple vehicles	Significant impact on operational performance, or noticeable operational impact for multiple vehicles
4	Life threatening or fatal injuries for multiple vehicles	Driver or vehicle tracking for multiple vehicles	Heavy financial losses for multiple vehicles	Significant operational impact for multiple vehicles.

risk levels based on threat, impact and level of security needed [27]. Its severity levels are similar to EVITA's, in both structure and content, and forms an alternative example of an automotive risk assessment framework.

EVITA and HEAVENS are both referenced in SAE J3061, the Cybersecurity Guidebook for Cyber-Physical Systems [28], which provides a framework of recommendations for establishing a cybersecurity engineering process within an organisation. The high-level framework for these process recommendations is aligned with that used in the automotive functional safety standard ISO 26262 [29], which is itself based on the well-established systems engineering V-model. Within that framework, J3061 suggests engineering processes that are suitable for security engineering, which among others include threat modelling, attack trees, vulnerability analysis and penetration testing. These key recommended process steps of J3061 are of particular relevance to the subject of this paper.

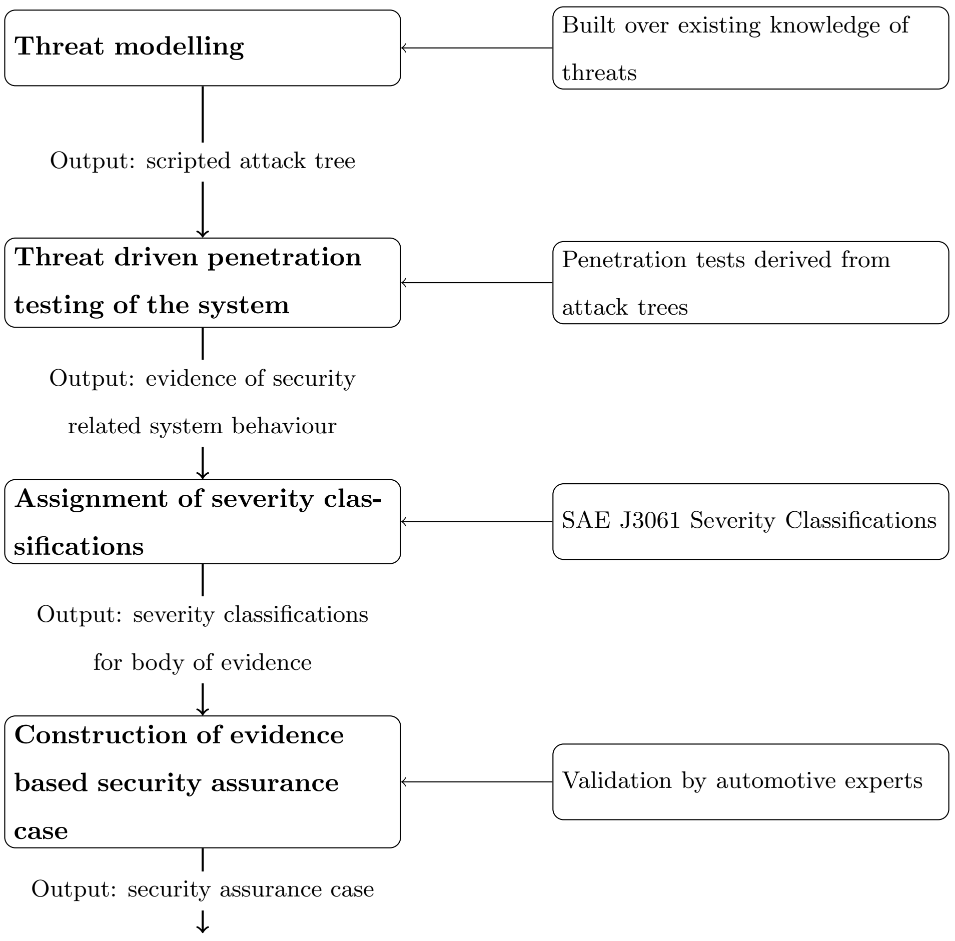
3. Methodology

We begin by describing our workflow in Section 3.1. We then discuss the methods involved in our workflow in subsequent sections (Sections 3.2 and 3.3) before reporting on our implementation of the process using Bluetooth as a case study (Section 3.4).

3.1. Workflow

The high level workflow can be seen in Figure 1. We expand on what the steps are, the methods used to achieve them and the outputs of each step in the following sections:

- In Section 3.1.1, we cover threat modelling, which we perform manually;
- In Section 3.1.2, we outline the semi-automated penetration testing process;
- In Section 3.1.3, we briefly describe the assignment of severity classifications (this is also semi-automated); and



Threat modelling
Built over existing knowledge of threats
Output: scripted attack tree
Threat driven penetration testing of the system
Penetration tests derived from attack trees
Output: evidence of security related system behaviour
Assignment of severity classifications
SAE J3061 Severity Classifications
Output: severity classifications for body of evidence
Construction of evidence based security assurance case
Validation by automotive experts
Output: security assurance case

Figure 1: Workflow of Methodology

- In section 3.1.4, we give details regarding the security assurance case construction.

3.1.1. Threat modelling

We perform threat modelling first, by creating attack trees based on prior knowledge and reconnaissance. This is a manual process based on expert judgment. The process is described in more detail in Section 3.2. The attack trees are captured in a particular format, which leads to the first output: a scripted attack tree that reflects possible penetration tests, and that also serves as input into the next step.

3.1.2. Threat driven penetration testing of the system

We then used penetration testing (as detailed in Section 3.3) using a semi-automated test execution tool (Section 3.4). The penetration tests are derived from the attack trees (see Figure 2); each test is a combination of the leaf nodes (attack steps) of the attack tree depending on the logic gate at the root of each branch. Results were acquired regarding system behaviour from a security perspective in a semi-automated fashion. Since this is an implementation-specific step, further details on methods used to test (i.e. gain evidence from) the case study interface of Bluetooth is given in Tables 2 and 3.

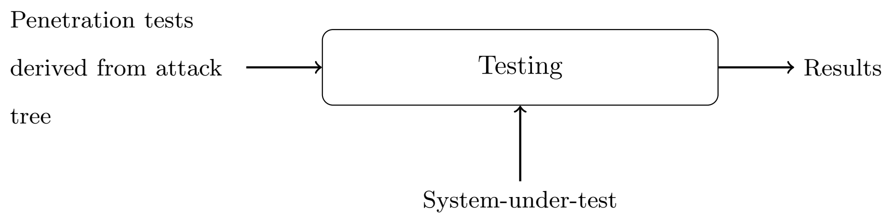


Figure 2: Penetration testing process

3.1.3. Assignment of severity classifications

The outcomes of systematic tests were assigned severity ratings (see Section 3.4.2 for case-study specific details). This assignment takes place based on several factors:

- Whether the result is a positive one (presence of information), or whether it was a negative one (no information acquired, but the test was run successfully)
- In the case of the latter, the assigned rating for each category is always zero. If the test was not possible, the assigned rating is also always zero, although rationale might differ.

- In the case of the former, questions are asked of the tester to record manual observations (see Section 3.4.2 for specific questions relating to Bluetooth). These guiding questions are based again on the rationale provided by EVITA in Table 1 and help form the ratings of 1, 2 or 3. The rating of 4 is used only in relation to multiple vehicles being affected at the same time; with none of the test vehicles being interconnected, the rating of 4 was never needed.

3.1.4. Construction of evidence based security assurance case

The construction of the assurance case is manual, but conforms to recommendations as outlined in SAE J3061.

SAE J3061 recommends that a cybersecurity case or assurance case is developed prior to release for production. The assurance case is analogous to a safety case and is used to document, through the provision of argumentation and a body of evidence, that a certain claim holds about the security achieved by the developed system [30]. An example of a claim could be: "freedom from unreasonable risk of an attack has been achieved"

This claim may be supported by a number of sub-claims which are further supported by arguments and associated evidence, such as the ones provided in our case studies in Section 5. Such a sub-claim could be based on the results of systematic security testing of the implementation, with the methods presented earlier used to acquire the evidence to support this sub-claim.

Therefore, having built our framework such that a systematic evaluation was possible, we also required a way to make the results of such an evaluation meaningful. In this paper, we use severity levels (as outlined in Table 1) to form ratings that could be used in a security assurance case.

Only the privacy (S_p) and operational (S_o) aspects of the EVITA classification scheme were considered. Safety (and safety analysis), being a significant topic in itself, was considered out of scope for this paper due to financial and informational constraints.

The financial category was also not considered due to two challenging factors. Firstly, the financial aspect could encompass the severity of loss through financial transactions within the vehicles, a feature that is not yet widely deployed and was not present on any of the vehicles tested. Secondly, the financial rating could also be due to severity of loss through theft of the vehicle. However, there is no definitive real world detection measure or tool to determine whether a vulnerability led to this theft, even should the vehicle be recovered.

Finally, having no information regarding internal paths or interfaces precludes us from formal verification and validation. There are also no set of expected outputs in the given context of automotive systems, since vehicular experimental analysis has typically concentrated on other technologies. Thus, this severity classification was validated by two domain experts.

3.2. Attack Trees

We use attack trees as our threat modelling method (see Section 3.1.1) to provide systematism. Attack trees are conceptual diagrams created to represent the actions of an adversary looking to fulfil an attack goal. These goals can be as low level (compromise an interface) or high level (steal money) as required. Logic gates connect the branches and leaves of the attack tree. The AND logic gate (also known as conjunction) requires that all leaves of a branch are complete before an attack is considered complete. The OR logic gate (also known as disjunction) requires that at least one of the leaves of a branch is complete before an attack is considered fulfilled [31]. Several other extended logic gates could also be used. In this study, sequential AND (SAND) is also used to denote an AND logic gate that requires temporal order. These trees can be represented as diagrams (Figure 3) or textually (Figure 4).

The trees used in this paper follow the broad outline (Figure 5) of the penetration testing process as described above. We populated the leaves of the attack trees manually with background research that had been performed on the case study interface Bluetooth (detailed in Section 3.4).

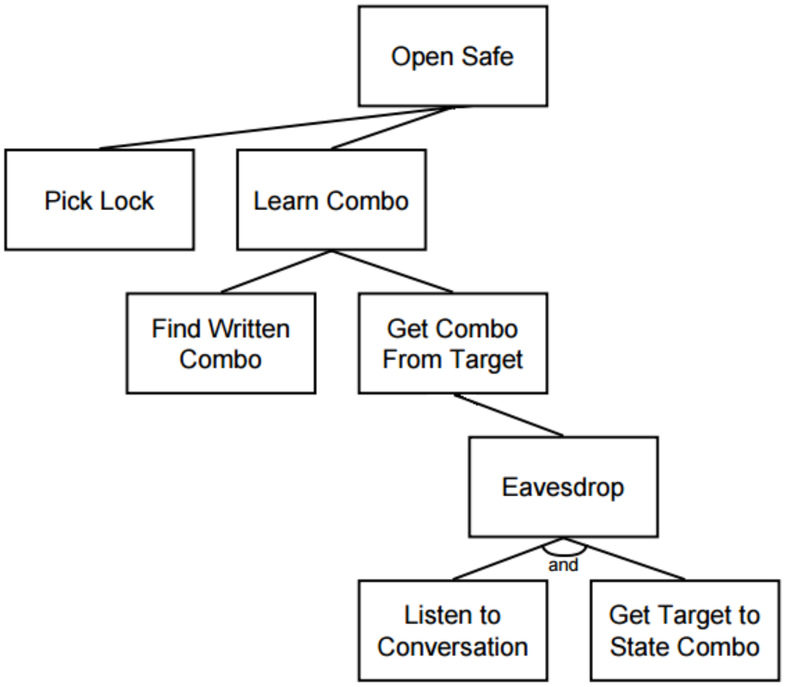
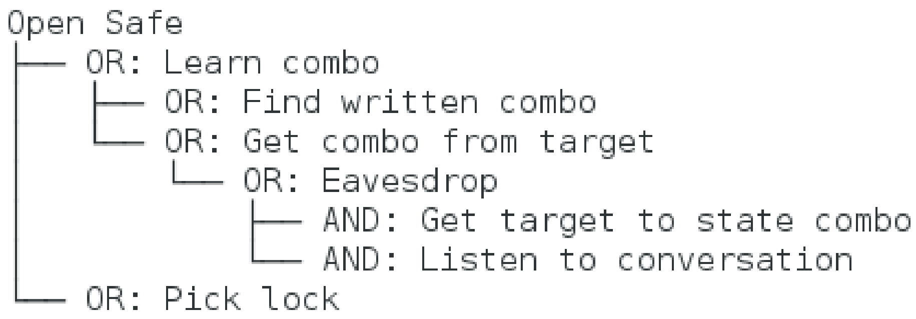


Figure 3: Example diagrammatic attack tree detailing the opening of a safe [11]



```
Open Safe
├── OR: Learn combo
│   ├── OR: Find written combo
│   └── OR: Get combo from target
│       └── OR: Eavesdrop
│           ├── AND: Get target to state combo
│           └── AND: Listen to conversation
└── OR: Pick lock
```

Figure 4: Example textual attack tree detailing the opening of a safe [11]

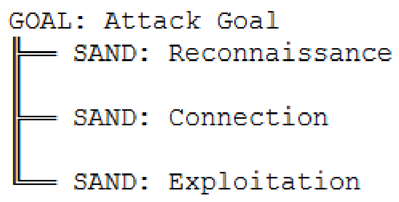


Figure 5: Board outline of attack trees used in this paper

3.3. Penetration Testing

This form of testing is a specialised form of security testing, and is the method used in the second step of our methodology (see Section 3.1.2). Ultimately, the aim of a such a test is to enumerate exploitable vulnerabilities and potential weaknesses depending on the scope and authorisation to carry out such a project. This is performed using a series of activities from an attacker's perspective. Activities are not usually prescribed, although there are methods and processes common in each test run regardless of the technology or system being tested (such as enumerating a network or machine address).

Although there are formal technical standards, there are drawbacks to applying them in this case. For example, ISO/IEC15408 has high informational needs about the testing environment, which is not suitable for an opaque system. Others (such as ISO/IEC 27001 on information security management) only address part of the whole challenge. Nevertheless, recognition of the fact that there had to be a broad common approach resulted in the proposal of the Penetration Testing Execution Standard (PTES) [32]. Although named as such, PTES is not a formal standard, but rather a technical methodological guideline to provide optimum test coverage; this, however, means that there is still a lack of a globally accepted methodology for penetration testing [11].

The advantage to these guidelines is that, although there is lack of a concrete consensus on what constitutes a flaw, weakness or vulnerability, the process can, at least, be loosely categorised and ordered around:

1. **Scoping**, whereby scope, objectives, aims and ground rules are established and agreed to between all stakeholder parties,

2. **Information gathering or reconnaissance**, which involves a background study on the test subject looking for all possible weaknesses, whether that be through user, developer or system documentation, publicly available manuals, source code or results of previous testing. This process can be passive (listening for information) or active (probing in order to acquire a response) depending on the aim of the test. An example of this could be profiling the Bluetooth module or chip, where the NAP and UAP information could be used to identify the manufacturer (a list is publicly available in IEEE's Standards Register [33]).
3. **Formation of vulnerability hypotheses**, which involves constructing hypotheses of possible vulnerabilities in the system, determined via study of background information gathered as well as by looking at abuse cases,
4. **Threat modelling** which involves the generation and modelling of possible threats based on potential vulnerabilities identified,
5. **Testing and exploitation**, which is used to establish the presence of the vulnerability, determining the nature of the flaw, whether it is repeated through the system and its security impact,
6. **Clean up and report**, which involves creating or collating recommendations for discovered flaws. These are then presented, along with a cleanup of the system to ensure that no inadvertent flaws from penetration testing (such as malware or backdoors) are left behind.

The categorisation above is not necessarily a step-by-step process. Each stage (or series of stages) can be re-iterated as needed for the system or component being tested; threat modelling for example may uncover a breadth of testing that might not necessarily be in scope, which would mean re-visiting or re-writing the scope in order to match any constraints more accurately.

Although a mass of information has been acquired regarding tools, techniques and to some extent, motivation (in terms of attacker goals and what they hope to gain), the issue of prioritisation and the combination of circumstance that would result in the use of this tool or that attack method can be covered by scenario building around an attack goal. In this paper, the latter is covered through the building of attack trees (see Section 3.2).

Of the steps above, scoping and information gathering as well as vulnerability hypotheses were inherent in the processes used to build the tailored attack trees that are seen later in experimental application (see Section 4). Direct reconnaissance of a specific implementation, testing, exploitation and reporting are embodied in the software implementation of the attack tree (see Section 3.4) and are reflected in its key features (Section 3.4.1).

3.4. Implementation

A software tool was designed and created to aid in the systematic evaluation process. This forms part of the threat driven penetration testing process as described in Section 3.1.2. This proof-of-concept tool was developed using Python 2.7, on a Kali Linux machine, with a Cambridge Silicon Radio Bluetooth 4.0 dongle attached. The tool incorporates and extends earlier Bluetooth testing tools [11]. The general architecture of the tool is given in Figure 6.

3.4.1. Key Features

As described above in Section 3.3, the features of the tool (and the methods which enable those features) can be categorised broadly into:

- **Reconnaissance**, which can be defined as a survey of the system's existence, configuration and capabilities (Table 2),
- **Connection attributes**, which includes information on pairing mechanisms, transmission sizes and connection state (Table 2), and
- **Attack goal**, which encompasses methods that would allow the realisation of the attack goal (Table 3);
- **Reporting**, including logs of all data gathering and tests run (Table 3), parameters chosen by the tester (where appropriate) and the severity classification. All are also for use as evidence in the security assurance case.

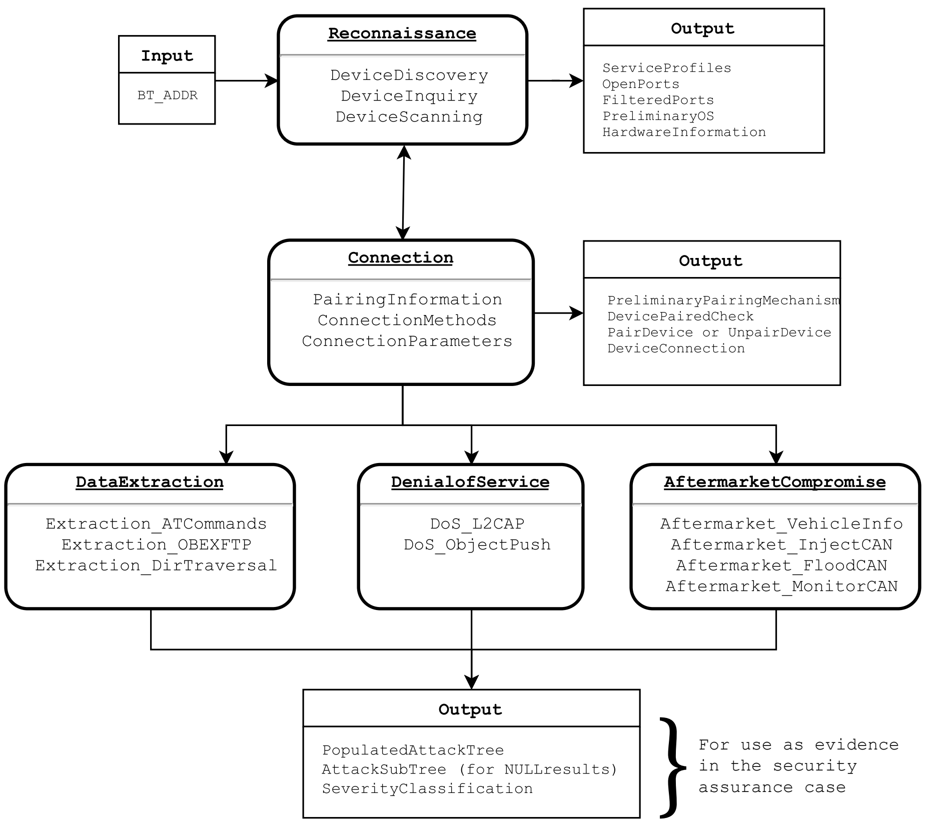


Figure 6: General architecture of the tool

Each of these categories is an individual component (with the individual attack steps forming sub-components). The attack goals in this case were two examples of the attack classification as described by [34] and are in line with the general accepted security testing goals of violating confidentiality, integrity or availability (CIA).

This structure allows for different permutations (depending on the attack tree desired), and for extensibility; new attack methods can be added as steps (as sub-components) within each module. New attack goals that come within test scope can also be added to the tool as a different module. Since the beginning of every security test begins with an inspection of the system, the reconnaissance module can likewise be re-used at the beginning of every test run.

Table 2: Proof-of-concept tool features (expanded version from [11])

	Feature	*Method*
Reconnaissance	(A) Discovery of 'discoverable' device addresses	Inquiry scans for available Bluetooth addresses
	(A) Discovery of 'hidden' devices	Brute-force scanning (incrementing the address bits by one before sending an inquiry). Requires pre-knowledge of the first three bytes of the Bluetooth address (OUI) - or other address bytes to be feasible.
	(A) Determination of device manufacturer	Using the OUI to scan through a database of stored OUIs (from IEEE's standard register [33])
	(A) Determination of Bluetooth chip manufacturer	Using device information supplied by *bluez*
	(S) Retrieves FCC ID information	Retrieves relevant Federal Communications Commission (FCC) web link if ID is known by user
	(A) Determination of service profiles offered by device	Using the Service Discovery Protocol (SDP)
	(A) Preliminary indication of device operating system (OS)	Using indicators in discovered service profiles
	(S) Determination of whether device uses legacy pairing	Checks Bluetooth version (version 2.0 or before means that legacy pairing is in use, whilst version 2.1 or above means that use of either legacy pairing or Secure Simple Pairing (SSP) is possible)
	(A) Determination of open ports	Sending information to all possible RFCOMM and L2CAP ports and awaiting the appropriate responses
	(A) Determination of filtered ports	Sending information to all RFCOMM and L2CAP ports and filtering for specific error messages
Connection Attributes	(A) Determination of pairing status	With reference to local paired devices list
	(S) Pair or unpair the device as appropriate	With reference to local paired devices list, subject to appropriate authentication
	(S) Spoof a device	Calls to installed *spooftooph*[35] package
	(A) Checks for presence of OBEX File Transfer Profile (FTP) service	With reference to discovered service profiles
	(A) Checks for presence of OBEX Push Profile (OPP) service	With reference to discovered service profiles
	(A) Determines maximum transmission unit (MTU) for open L2CAP ports	Sending increasing size of packets until Bluetooth error `90, message too long` appears

(A) = fully automated, (S) = semi automated, requires manual intervention

Table 3: Proof-of-concept tool features (continued) (expanded version from [11])

		Feature	*Method*
Attack Goal	*Data Extraction*	(S) Attempted extraction of information from headunit	Using "attention modem" (`AT`) commands through open RFCOMM ports
		(S) Attempted extraction of information by browsing headunit filesystem	Mounting the filesystem on a "Filesystem in Userspace" (FUSE) based filesystem type (if OBEX FTP exists)
		(S) Attempted extraction of information	Using the *dot-dot-slash* (`../`) attack to attempt directory traversal beyond the given restricted directory (if OBEX FTP exists)
	DoS	(A) Attempted denial of service	Flooding open L2CAP ports with L2CAP echo requests
		(S) Attempted denial of service	Repeated data push through OBEX channels
		(S) Attempted denial of service	Pushing of malformed data through any RFCOMM channels
	Vehicle Compromise	(A) Extract vehicle specific information	Vehicle information based on `AT` commands sent to an attached wireless Bluetooth-enabled OBD-II device
		(S) Attempted extraction or denial of service	Through injection or flooding using OBD-II protocol messages (see Section 4.2). User specifies parameters such as type and number of messages and time intervals
		(S) Attempted vehicle compromise	Through injection or flooding using raw predetermined CAN messages (see Section 4.2). User specifies CAN header ID, CAN data payload, number of messages to be sent and time intervals
		(A) Vehicle data extraction	Passive monitoring of all exposed CAN buses on the OBD-II port
Outputs		(A) Scan logs	Written to CSV or TXT files and collated at the end of the test run
		(A) Populated attack tree	Displayed and logged with results of the test run
		(A) Subtrees	Where test results have not been found or entered, denoted by `NULL`, appropriate subtrees (found using width-first search) will be displayed and logged.
		(S) EVITA Severity classifications	For each result in combination with answers to tester queries, a severity classification based on the "privacy" and "operational" categories (see Section 2.1) is given.

(A) = fully automated, (S) = semi automated, requires manual intervention

3.4.2. Severity Classification

After tests are performed, questions were asked of the tester to record manual observations. Note that these are to assist in the severity classifications (as outlined in Section 3.1.3). Although the name and characteristics of the service profiles are determined in an automated manner, a human may be required to categorise. For example, categories such as ”Vendor Specific Profiles” would require a human to assess with background research whether they were indeed specific to the target system vendor, or whether they were instead specific to a platform such as Android Auto, or to the Bluetooth version implemented on a target system.

These questions include:

- Service profiles and its nature:
    - Were there profiles that are named suggestively?
    - Were there vendor specific profiles?
    - Were there synchronisation profiles?
    - Were there Personal Ad-Hoc Network (PAN) profiles?

  Certain service profiles mean that personal information can be synchronised between two devices, which might impact privacy. Others may offer more access to the system itself, either by broadening the attack surface (such as by allowing Ethernet packets through e.g. through the PAN profile) or bespoke services which may have implementation flaws. Suggestive profiles (such as “Reflash Server” may help narrow the scope or provide a target for an adversary. These affect the privacy rating and potentially the operational rating.

- PIN behaviour:
    - Was the PIN dynamic?

  - Was the PIN customisable by user?
  - Was the PIN easily guessable?

  Each of these questions (depending on a positive or negative answer) would affect the risk of an adversary being able to compromise the Bluetooth connection through, for example, eavesdropping. In the worst case scenario of a static, fixed and easily-guessable PIN, the risk would be far greater than if the PIN had been dynamic or customisable. This would affect both operational and privacy ratings.

- Data returned by the vehicle:
  - Was there information about the vehicle returned?
  - Was there information about the user of the vehicle returned?

  These questions would affect the privacy ratings given, with the latter being the highest severity.

- The behaviour of the vehicle during testing:
  - Was there discernible operational impact on the system during testing?
  - Was there a reaction on the user interface?

These questions were both used to discern the impact on the system, and whether any alerting effects resulted. This affects the operational rating during classification.

The combination of answers were used in conjunction with the findings of the tests to generate tabulated ratings for each aspect of the test suite. Ratings given to each of the test findings are labelled 'ACT' (for actual). Furthermore, a potential severity level is also given (denoted by 'POT').

The rationale for ACT and POT ratings is hard-coded into the tool. This is possible for the ACTUAL ratings, since EVITA provides concrete and discrete rationale for each rating.

The POT ratings are based on worst-case scenarios should an exploit or attack be possible, or if a part of the system is inherently weak. An example of this is an unchangeable short PIN, which in the worst case would allow an attacker to compromise authentication with all the attendant risks. All are also tied to the rationale outlined by EVITA.

Each rating is given as $\mathrm{S_p}$i, $\mathrm{S_o}$i where $\mathrm{S_p}$ represents the privacy severity rating, and $\mathrm{S_o}$ the operational severity rating and where $i \in \{0-4\}$.

The tool also checks through the tree to give automatic ratings. For example, where there is no OBEX FTP, the mounting and traversal attacks were not performed. This would automatically result in an $\mathrm{S_p}0$ and $\mathrm{S_o}0$ rating. Other aspects such as vehicle tracking through the Bluetooth address (which affects privacy), and whether there were open ports (which potentially affects operations) are pulled from the logs created on the findings and and also classified automatically.

3.5. Extension of Methodology

We have used part of the methodology described above in our earlier work [11]. This includes the threat modelling aspect (outlined in Section 3.1.1) and the penetration testing aspect (Section 3.1.2). In this paper, we have expanded in the breadth of application by:

- Expanding on the number of attack trees. This is synonymous with enumerating more attack goals. In this paper we add Denial of Service and Vehicle Compromise, embodied in a software tool (see Section 3.4) ;
- Adding an additional vector through which vehicle compromise could take place (i.e. compromise through the diagnostics port using an aftermarket device) (see Section 4.2);

We have also extended the workflow of the methodology itself by:

- Adding to the methodology the assignment of severity levels to results of the systematic evaluations. The severity level assignment process and how this could contribute to a security assurance case is also discussed above (Sections 3.1.3 and 3.1.4).

- Experimental application of this new step in the workflow. This is described in Section 5.

4. Experimental Application

We use the Bluetooth interface as a case study for our experimental application. The reason for this is its ubiquity in automotive systems, whether that be in-cabin (via the infotainment system) or through aftermarket devices that attach to the vehicle for diagnostics. All features of the toolset as described in Tables 2 and Tables 3 were used to acquire the data as given below. The exception to this is where a specific implementation is required (such as the presence of an OBEXFTP service), and these are marked in the experimental results in the following sections.

4.1. Experimental Methodology – Infotainment System

Two attack trees (Figure 7 and Figure 8) with the attack goals of data extraction and denial of service were predetermined using Bluetooth techniques and attacks described in literature.

Vehicles were stationary and ignition was switched on, ensuring that it was within the Class 2 (ten metre) range as no antennas were used to extend range. If Bluetooth had to be enabled, then this was performed before the tool was run. The vehicle was a small hatchback from a major manufacturer registered in 2013.

For this study, primarily "legitimate" connections were used, with legitimate defined here as a straightforward connection to the vehicle from a mobile phone, without any attempt to compromise the Bluetooth communications protocol. The reason for this was that the aim of the study was to explore Bluetooth as implemented on the vehicular system, rather than the Bluetooth protocol itself, since the security (or lack thereof) of the Bluetooth pairing mechanisms and the underlying protocols are well studied [36, 37, 38, 39]. A connection compromise would thus exacerbate the risk of malicious actions rather than introduce a new risk. The latter is especially significant since there are no access control policies on the vehicle, i.e. access to Bluetooth services implemented was not differentiated by user, but rather by device. Every device had access to every service (assuming compatibility, which is a different issue). There was no need to compromise the protocol to elevate privilege or gain access to restricted areas as might be the case with a more traditional computing system. The single caveat to this was the use of a spoofed device (at the local testing interface), to observe whether the vehicle accepted the connection.

4.1.1. Attack goals

Two attack trees (according to two attack goals) were created for testing on vehicles. They are discussed below.

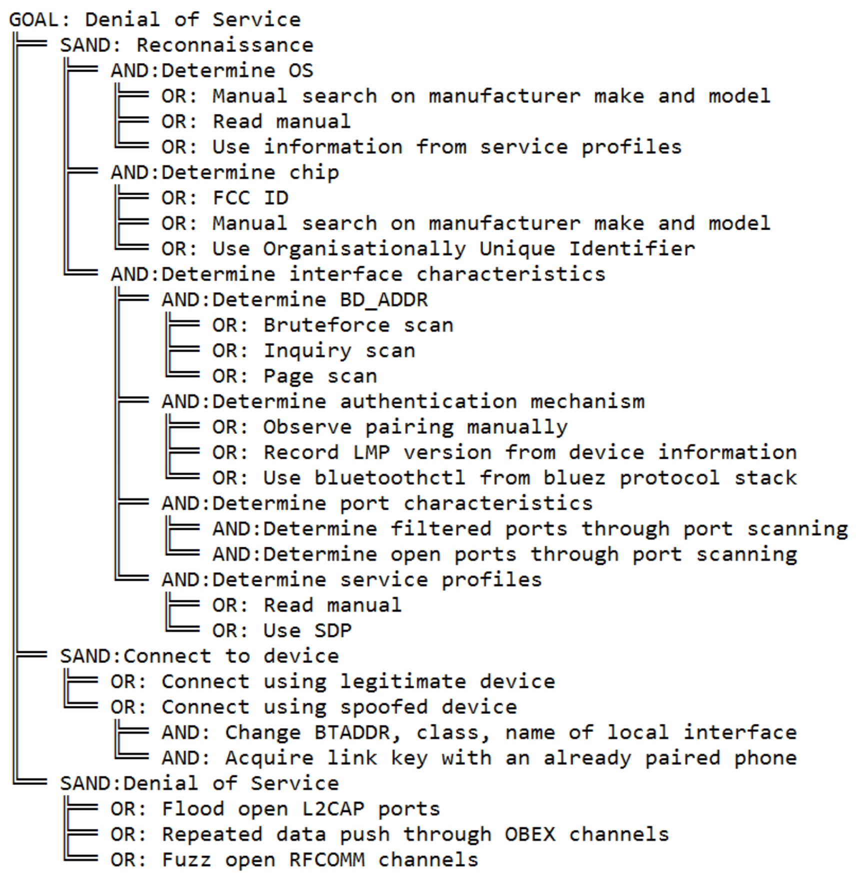
Data extraction. from a vehicle through Bluetooth is largely an exercise in gathering as much information as possible. Although the most overtly valuable is information regarding the user of the vehicle, data from the vehicle itself (such as cornering speed, braking times and so forth) can be used to fingerprint drivers [40].

Other information that may be of use include the age of the technology (which may indicate legacy flaws), the chip manufacturer (which may point to analogous vulnerabilities or bugs), pairing mechanisms (where aspects such as using a fixed PIN might make a system more insecure) or other reconnaissance data that could enable targeted attacks. The attack tree used for this attack goal is shown in Figure 7.

Denial of Service. involves flooding and fuzzing, as jammers were not available at the time of testing. The aim was to cause disruption to the vehicle or any component therein, with the primary target being the headunit. Because of the nature of denial of service, and because the system under test is a black box, most of the results are based on observation whilst trying to perform normal

```
GOAL: Data extraction
╠══ SAND: Reconnaissance
║   ╠══ AND:Determine OS
║   ║   ╠══ OR: Manual search on manufacturer make and model
║   ║   ╠══ OR: Read manual
║   ║   ╚══ OR: Use information from service profiles
║   ╠══ AND:Determine chip
║   ║   ╠══ OR: FCC ID
║   ║   ╠══ OR: Manual search on manufacturer make and model
║   ║   ╚══ OR: Use Organisationally Unique Identifier
║   ╚══ AND:Determine interface characteristics
║       ╠══ AND:Determine BD_ADDR
║       ║   ╠══ OR: Bruteforce scan
║       ║   ╠══ OR: Inquiry scan
║       ║   ╚══ OR: Page scan
║       ╠══ AND:Determine authentication mechanism
║       ║   ╠══ OR: Observe pairing manually
║       ║   ╠══ OR: Record LMP version from device information
║       ║   ╚══ OR: Use bluetoothctl from bluez protocol stack
║       ╠══ AND:Determine port characteristics
║       ║   ╠══ AND:Determine filtered ports through port scanning
║       ║   ╚══ AND:Determine open ports through port scanning
║       ╚══ AND:Determine service profiles
║           ╠══ OR: Read manual
║           ╚══ OR: Use SDP
╠══ SAND:Connect to device
║   ╠══ OR: Connect using legitimate device
║   ╚══ OR: Connect using spoofed device
║       ╠══ AND: Change BTADDR, class, name of local interface
║       ╚══ AND: Acquire link key with an already paired phone
╚══ SAND:Extract data
    ╠══ OR: Use AT commands
    ╠══ OR: Use OBEXFTP (directorytraverse)
    ╚══ OR: Use OBEXFTP (filemount)
```

Figure 7: Attack tree with data extraction as an attack goal (adapted from [11])



```
GOAL: Denial of Service
╠══ SAND: Reconnaissance
║   ╠══ AND:Determine OS
║   ║   ╠══ OR: Manual search on manufacturer make and model
║   ║   ╠══ OR: Read manual
║   ║   ╚══ OR: Use information from service profiles
║   ╠══ AND:Determine chip
║   ║   ╠══ OR: FCC ID
║   ║   ╠══ OR: Manual search on manufacturer make and model
║   ║   ╚══ OR: Use Organisationally Unique Identifier
║   ╚══ AND:Determine interface characteristics
║       ╠══ AND:Determine BD_ADDR
║       ║   ╠══ OR: Bruteforce scan
║       ║   ╠══ OR: Inquiry scan
║       ║   ╚══ OR: Page scan
║       ╠══ AND:Determine authentication mechanism
║       ║   ╠══ OR: Observe pairing manually
║       ║   ╠══ OR: Record LMP version from device information
║       ║   ╚══ OR: Use bluetoothctl from bluez protocol stack
║       ╠══ AND:Determine port characteristics
║       ║   ╠══ AND:Determine filtered ports through port scanning
║       ║   ╚══ AND:Determine open ports through port scanning
║       ╚══ AND:Determine service profiles
║           ╠══ OR: Read manual
║           ╚══ OR: Use SDP
╠══ SAND:Connect to device
║   ╠══ OR: Connect using legitimate device
║   ╚══ OR: Connect using spoofed device
║       ╠══ AND: Change BTADDR, class, name of local interface
║       ╚══ AND: Acquire link key with an already paired phone
╚══ SAND:Denial of Service
    ╠══ OR: Flood open L2CAP ports
    ╠══ OR: Repeated data push through OBEX channels
    ╚══ OR: Fuzz open RFCOMM channels
```

Figure 8: Attack tree with denial of service as an attack goal

actions (such as making a phone call), or on what might happen on the graphical front end, rather than anything being returned by the vehicle. The attack tree used for denial of service is shown in Figure 8.

Results from applying the tool in accordance with both the attack trees as given above, as well as implications of these results is given in Table 4.

4.2. Experimental Methodology – Aftermarket Device

The experimental application of the toolset involves the ”Vehicle Compromise” attack goal suite as seen in Table 3. The four attempted data extraction or vehicle compromise tools correspond to the final `SAND:Cause Vehicle`

Table 4: Experimental Results: Vehicle 1 [11]

	Interface characteristics	*Observation*
	Address: xx:xx:xx:34:8A:2D **Version:** 2.0 **Class:** 0x340408 **Services:** HFP, SyncML Server, A2DP, AVRCP, PBAP (Client), OBEX OPP, MAP MNS **Open ports:** RFCOMM 1,4 and L2CAP 1,3,23,25	Bluetooth version 2.0 means that vehicle is using legacy pairing exclusively. Vehicle produces dynamic 6 digit PIN. Device class interprets to an audio/video hands-free device Services include a Synchronization Markup Language (SyncML) Server (for phonebook, message and calendar information synchronisation). A SyncML client could be used to extract personal information that is stored on the vehicle (although connections through this were unstable - no information was found).
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	Responds to all `AT` commands sent on channel 4 with `AT+BRSF=39`. Commands on other channels end with errors such as `103, Software caused connection abort`. No OBEXFTP, so filesystem cannot be accessed through this vector.
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Two responsive L2CAP ports found, with maximum transmission unit (MTU) size of 4096 and 242 respectively. Flooding of the port with the larger MTU resulted in discoverable mode disrupted intermittently when trying to pair. Calls made had quality issues or were dropped.

Additional observations
Time discoverable was limited to two minutes, and the user had to enable Bluetooth. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any of the attempted actions beyond pairing. The vehicle recognised a spoofed device as one that has previously paired, although authentication checks failed (probably due to incorrect location on the test laptop of the link key acquired from a previously paired device).

1-bit	11-bit	1-bit	1-bit	1-bit	1-bit	4-bit	$\leq$ 64-bit	16-bit	2-bit	7-bit	7-bit
SOF	IDE	RTR	IDE	r1	r0	DLC	Data	CRC	ACK	EOF	IFS

Figure 9: CAN frame format [41]

`Compromise` branch in the attack tree as described in Figure 10.

Typically, messages that are sent into the OBD port are either raw Controller Area Network (CAN) or diagnostic messages. These two areas of intra-vehicular communications are introduced below.

4.2.1. CAN Messages

The CAN protocol is the primary mode of communication inside the vehicle. The latest version is CAN 2.0, first specified in 1991 [41] and embodied as an ISO standard (ISO11898) in 2003. These CAN messages carry much of the information needed for the operation and control of the vehicle.

The standard CAN packet comprises (up to) 11 bits for the message ID, followed by (up to) 8 bytes of data, then a cyclic redundancy check (16 bits) for error detection. The full frame format is given in Figure 9, with descriptions of each segment in Table 5. The extended CAN frame format uses 29 bits instead for the message ID with slightly different configurations of bits to allow for this. We concentrate here on the standard CAN message only. The full 8 bytes of data need not be used. Information for a door sensor, for example, may only require 1 bit. Conversely a message can be spread across many frames, with various data lengths and offsets.

Arbitration, should nodes on the CAN network transmit simultaneously, is based on message prioritisation. This prioritisation is determined using the message ID, with the lowest ID being the highest priority. That being the case, implementation usually means that mission-critical messages are the ones assigned lower IDs.

Assignment of IDs along with data payload is manufacturer specific, however, reuse is common to save on the cost of redesigning a network [2]. Although CAN data is not typically encrypted, reverse engineering can be a difficult pro-

Acronym	*Description*
SOF	Start of File
IDE	The Identifier Extension establishes the priority of the message. The lower the binary value of the ID, the higher its priority. A CAN message frame with an 11-bit ID is a *standard* frame. One with a 29-bit ID is an *extended* frame.
RTR	Remote Transmission Request; if this bit is dominant (i.e. 0), more information is necessary from another node
r0/r1	Reserved bits originally, but now in use in some implementations to identify XOR masked CAN messages
DLC	The Data Length Code contains the number of data bytes to be transmitted
CRC	Cyclic Redundancy Check for error detection
ACK	The Acknowledge bit is overwritten (from recessive (1) to dominant (0)) to acknowledge validity
EOF	End of File
IFS	The interframe space contains time required to move a received frame to the message buffer area

Table 5: CAN frame format descriptions [41]

cess considering the volume and variety of content that is transmitted. This is especially the case without a CAN database, which contains definitions for every message and signal. This file is often highly confidential. However, specific CAN messages for discrete events (such as unlocking doors) can be obtained in a relatively straightforward manner through trial and error experiments.

CAN data is transmitted on a CAN network (in a bus configuration). Therefore any Electronic Control Unit (ECU) on the network has access to all messages. There is no addressing; instead each ECU is programmed to listen to a set of CAN IDs, which triggers some pre-determined functionality.

4.2.2. Diagnostic Messages

Parameter IDs (PIDs) are used to perform diagnostic functions or request data from the vehicle specifically through the OBD-II port. This is done through a query-response mechanism, where a PID query comprises the CAN ID `7DF`, followed by 8 data bytes. The first byte is the data length (usually 02) with the second byte being the *mode* and the third byte typically being the PID. The combination of modes and PIDs can then be sent into the CAN bus, and a response should be received from whatever in-vehicle module is responsible (if any). The response CAN ID is typically 8 (in hex) higher than the message ID that the responding ECU answers to. A response of `NO DATA` usually indicates that the vehicle has not returned anything, a response beginning with `7F` in byte 2 means that the vehicle does not recognise the request.

The first ten modes (`01` to `0A`, described in SAE J1979 (E/E Diagnostic Test Modes) [42], are standard and generic to all compliant vehicles. In these standard modes, the PID is only the 2nd byte, with the 3rd to 8th byte unused. With non-standard modes, the PIDs could extend to the 3rd byte. Manufacturers, are not obliged to implement all standard commands, and additionally could also define functions for non-standard PIDs. There is much information that could be gathered using PIDs to interrogate the vehicle. For example, sending the mode `09` with PID `02` retrieves the Vehicle Identification Number (VIN). The VIN is unique to the vehicle and is used for many activities, from vehicle maintenance to recovery of a stolen vehicle.

Another set of (related) diagnostic messages called Unified Diagnostic Service (UDS) messages are embodied in ISO 14229-1 (Road Vehicles - Unified Diagnostic Services) [43]. This standard specifies the requirements for diagnostic services independent of the data link connection between vehicle and remote device.

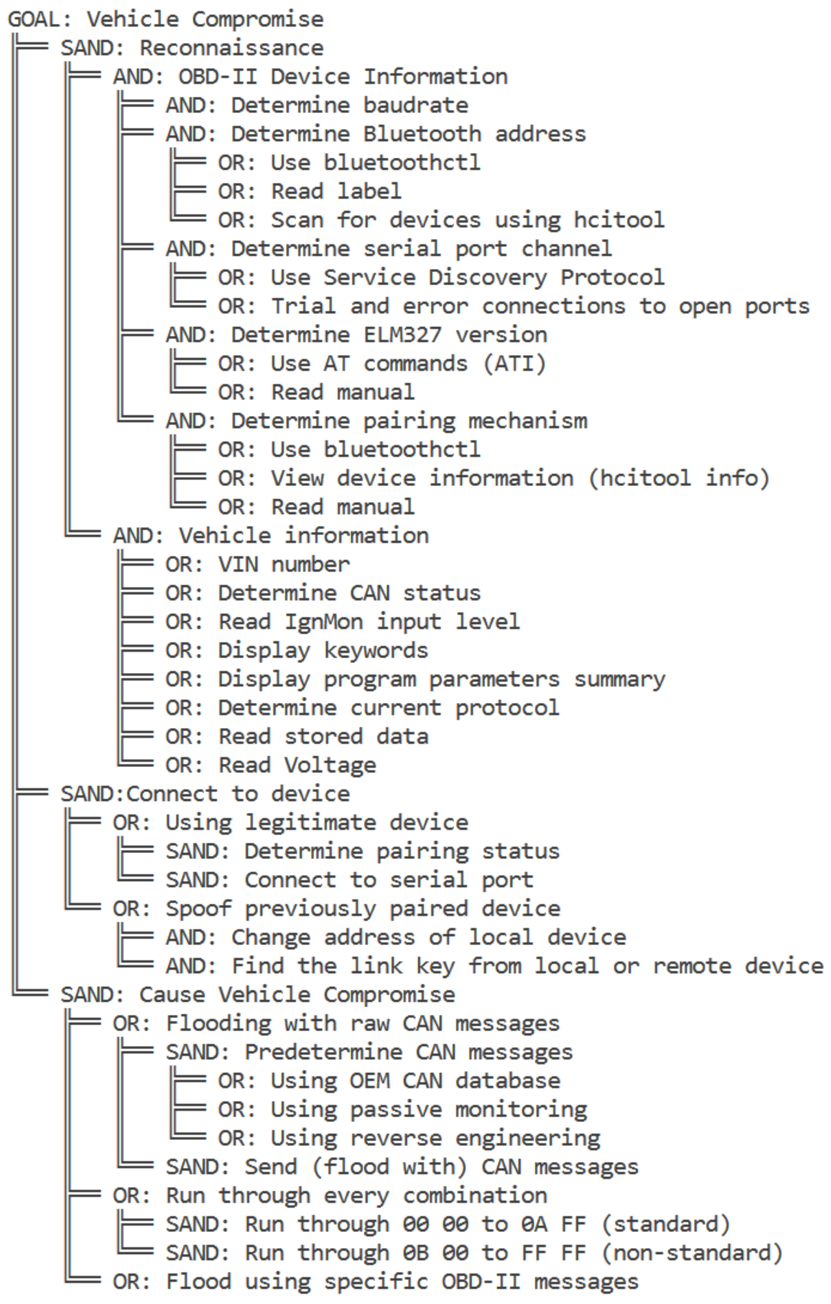
Like the J1979 OBD-II messages, UDS works to a request-response system. Particular service IDs (SIDs) are sent to the vehicle (more specifically to ECUs that support a particular service) in order to trigger a pre-determined functionality, whether that be to start a Diagnostic Management Session, interrogate UDS-compatible ECUs or reset the ECU. Again, although the standard determines what some of the UDS messages do (such as the ones given in the examples above), manufacturers are able to define their own SIDs.

4.2.3. Setup

We concentrate on what combinations of diagnostic or CAN messages might cause a reaction. We describe below the setup of the systematic evaluation of a vehicle through an attached Bluetooth-enabled OBD-II device.

The OBDLINK MX dongle was connected to a single test vehicle. This particular device was chosen as we could be sure that the chip was not counterfeit and able to accept the full `AT` command set (a full list of supported commands can be found in ELM's `AT` command list [44]). Physical setup was the same as that of the case study. From here the proof-of-concept tool used a pre-determined attack tree (shown in Figure 10) to run through the entire aftermarket device test suite.

Baudrate was set at 115200, which was the maximum based on the ELM device (with the default connection being set at 9600). Time intervals for all messages was set at 0.5 seconds in order to ensure that the OBD-II device had time to read and transmit the data, although this is user customisable using the proof-of-concept tool. Although a serial connection is slow compared to the speed CAN busses could operate at (40Kbit/s to 1Mbit/s for high-speed CAN



```
GOAL: Vehicle Compromise
╞══ SAND: Reconnaissance
│   ╞══ AND: OBD-II Device Information
│   │   ╞══ AND: Determine baudrate
│   │   ╞══ AND: Determine Bluetooth address
│   │   │   ╞══ OR: Use bluetoothctl
│   │   │   ╞══ OR: Read label
│   │   │   ╘══ OR: Scan for devices using hcitool
│   │   ╞══ AND: Determine serial port channel
│   │   │   ╞══ OR: Use Service Discovery Protocol
│   │   │   ╘══ OR: Trial and error connections to open ports
│   │   ╞══ AND: Determine ELM327 version
│   │   │   ╞══ OR: Use AT commands (ATI)
│   │   │   ╘══ OR: Read manual
│   │   ╘══ AND: Determine pairing mechanism
│   │       ╞══ OR: Use bluetoothctl
│   │       ╞══ OR: View device information (hcitool info)
│   │       ╘══ OR: Read manual
│   ╘══ AND: Vehicle information
│       ╞══ OR: VIN number
│       ╞══ OR: Determine CAN status
│       ╞══ OR: Read IgnMon input level
│       ╞══ OR: Display keywords
│       ╞══ OR: Display program parameters summary
│       ╞══ OR: Determine current protocol
│       ╞══ OR: Read stored data
│       ╘══ OR: Read Voltage
╞══ SAND:Connect to device
│   ╞══ OR: Using legitimate device
│   │   ╞══ SAND: Determine pairing status
│   │   ╘══ SAND: Connect to serial port
│   ╘══ OR: Spoof previously paired device
│       ╞══ AND: Change address of local device
│       ╘══ AND: Find the link key from local or remote device
╘══ SAND: Cause Vehicle Compromise
    ╞══ OR: Flooding with raw CAN messages
    │   ╞══ SAND: Predetermine CAN messages
    │   │   ╞══ OR: Using OEM CAN database
    │   │   ╞══ OR: Using passive monitoring
    │   │   ╘══ OR: Using reverse engineering
    │   ╘══ SAND: Send (flood with) CAN messages
    ╞══ OR: Run through every combination
    │   ╞══ SAND: Run through 00 00 to 0A FF (standard)
    │   ╘══ SAND: Run through 0B 00 to FF FF (non-standard)
    ╘══ OR: Flood using specific OBD-II messages
```

Figure 10: Attack tree used to test vehicles which have an aftermarket OBD-II device attached

for example), the set interval was enough to be able to flood the bus with enough messages to cause adverse reactions. Lower level bit-by-bit attacks, however, would not be feasible using this method.

The systematic test was first performed with ignition on (but not engine), with the assumption that, as long as the appropriate target ECUs were powered, that the vehicle would respond. Deciphering the content of the response was considered out of scope at this point in time, as we had no manufacturer CAN database available to interpret the CAN data acquired from the bus. The experiments were then repeated with the engine on with the modes and PIDs that either returned CAN data or caused a physical reaction from the vehicle.

Of the standard suite of modes, the vehicle returned information from in three modes [42]:

- `01`, which corresponds to "Show current data";
- `06`, which corresponds to "Test results, oxygen sensor monitoring for CAN only" and
- `09`, which corresponds to "Request vehicle information"

Of the non-standard suite of modes, the vehicle returned information for nine different modes with all other modes returning `NO DATA`. Note that there were modes and PIDs where the vehicle returned `NO DATA`, but that there was a physical effect on the vehicle.

A summary of results can be seen in Table 6. Because non-standard modes are manufacturer dependent, the exact modes and PIDs found to affect the vehicle are not given.

Select raw CAN messages were also sent into the vehicle. These packets were pre-determined through trial and error, but consisted of messages that were known to cause an effect when trialled through a wired connection to the OBD-II port. Unlike the diagnostic messages, some of the CAN messages sent through only needed to be sent once. A summary of these results can be seen in Table 7.

Table 6: Results of systematic testing (Modes and PIDs)

Modes found	*Result*	*Description*
First mode	CAN data returned for 2 PIDs	The first PID caused the headunit screen to display "Diagnostics Mode On", the second PID caused the engine to refuse to start and the hazard lights (on the cluster only) to flash, but required message flooding. If engine is on, this causes the engine to stall. Vehicle remains unresponsive thereafter as long as message flooding continues. Once flooding stops, the instrument cluster restarts but not the engine.
Second mode	CAN data returned for 1 PID	The first PID had no physical effect, the second returned `NO DATA`, but the electronics cut out, with the instrument panel and ignition button non-responsive.
Third mode	Not recognised	`7F` returned by vehicle
Fourth mode	CAN data returned for many PIDs	12 of the PIDs each had 16 frames worth of CAN data returned by the vehicle
Fifth mode	Not recognised	`7F` returned by vehicle
Sixth to ninth mode	Not recognised	`7F` returned by vehicle

Table 7: Results of systematic testing (raw CAN messages)

Action	*Observation*
Unlock doors	Vehicle did not return any data, doors did not unlock, both with key in-cabin or external to the vehicle
Sending Unified Diagnostic Service messages	Hazard lights came on, “crash” was displayed on the secondary screen above the steering wheel. Vehicle doors continuously locked and unlocked. The former happened if message is sent once, the latter if message flooding is performed
Changing speed and RPM indicators on the instrument panel	CAN message flooding to this particular message ID caused the needles to fluctuate
Disabling power steering	Successful, but only if the vehicle was stationary. This message only needed to be sent once. Further message flooding had no effect.

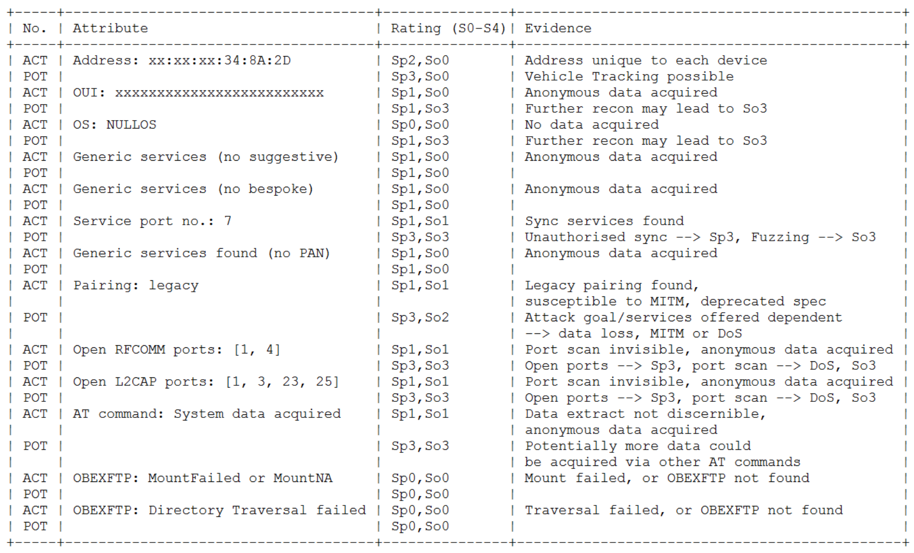
5. The Security Assurance Case

In this section, we discuss the classification of the results obtained (i.e. the evidence) through systematic evaluations and the assignment of severity ratings to this evidence. Methods used to acquire this evidence (i.e. test results) are given in Tables 2 and 3. This follows the methodology as given in Section 3, using the implementation as described in Section 3.4.2.

Recall that only the privacy and operational aspects of the EVITA classification scheme (see Section 3.4.2) were considered. The assignment of ratings were based on the EVITA severity ratings (Section 2.2) and a rationale is given for each of the ratings given.

5.1. Infotainment system

The severity ratings that were assigned for the data extraction and denial of service attack goals are given in Figure 11 and Figure 12 respectively.



```
+-----+------------------------------------+-------------+---------------------------------------------+
| No. | Attribute                          | Rating (S0-S4)| Evidence                                  |
+-----+------------------------------------+-------------+---------------------------------------------+
| ACT | Address: xx:xx:xx:34:8A:2D         | Sp2,So0     | Address unique to each device               |
| POT |                                    | Sp3,So0     | Vehicle Tracking possible                   |
| ACT | OUI: xxxxxxxxxxxxxxxxxxxxxxxx      | Sp1,So0     | Anonymous data acquired                     |
| POT |                                    | Sp1,So3     | Further recon may lead to So3               |
| ACT | OS: NULLOS                         | Sp0,So0     | No data acquired                            |
| POT |                                    | Sp1,So3     | Further recon may lead to So3               |
| ACT | Generic services (no suggestive)   | Sp1,So0     | Anonymous data acquired                     |
| POT |                                    | Sp1,So0     |                                             |
| ACT | Generic services (no bespoke)      | Sp1,So0     | Anonymous data acquired                     |
| POT |                                    | Sp1,So0     |                                             |
| ACT | Service port no.: 7                | Sp1,So1     | Sync services found                         |
| POT |                                    | Sp3,So3     | Unauthorised sync --> Sp3, Fuzzing --> So3  |
| ACT | Generic services found (no PAN)    | Sp1,So0     | Anonymous data acquired                     |
| POT |                                    | Sp1,So0     |                                             |
| ACT | Pairing: legacy                    | Sp1,So1     | Legacy pairing found,                       |
|     |                                    |             | susceptible to MITM, deprecated spec        |
| POT |                                    | Sp3,So2     | Attack goal/services offered dependent      |
|     |                                    |             | --> data loss, MITM or DoS                  |
| ACT | Open RFCOMM ports: [1, 4]          | Sp1,So1     | Port scan invisible, anonymous data acquired |
| POT |                                    | Sp3,So3     | Open ports --> Sp3, port scan --> DoS, So3  |
| ACT | Open L2CAP ports: [1, 3, 23, 25]   | Sp1,So1     | Port scan invisible, anonymous data acquired |
| POT |                                    | Sp3,So3     | Open ports --> Sp3, port scan --> DoS, So3  |
| ACT | AT command: System data acquired   | Sp1,So1     | Data extract not discernible,               |
|     |                                    |             | anonymous data acquired                     |
| POT |                                    | Sp3,So3     | Potentially more data could                 |
|     |                                    |             | be acquired via other AT commands           |
| ACT | OBEXFTP: MountFailed or MountNA    | Sp0,So0     | Mount failed, or OBEXFTP not found          |
| POT |                                    | Sp0,So0     |                                             |
| ACT | OBEXFTP: Directory Traversal failed | Sp0,So0    | Traversal failed, or OBEXFTP not found      |
| POT |                                    | Sp0,So0     |                                             |
+-----+------------------------------------+-------------+---------------------------------------------+
```

Figure 11: Severity classification ratings for the data extraction test suite

The address, as a unique identifying factor of the headunit, could also be used to track a vehicle. In terms of the Organisationally Unique Identity (OUI) (which is the first three bytes of the Bluetooth address registered to a specific manufacturer via IEEE), the fact that this organisation is known could then lead to further reconnaissance (including research into known bugs or software defects), which could lead to significant impairment. The operating system could not be enumerated by the tool in this case, and so an automatic rating of 0 on both privacy and operational fronts was assigned. This could be adjusted based on the results of manual observation.

The services discovered were all generic (no suggestive or bespoke services), with no PAN services identified. This is based on tester observation (see Section 3.4.2), although could be further automated based on a scan of the log produced from the initial reconnaissance stage. This is all anonymous data (S_p1), but operationally has no impact. Legacy pairing was identified, and so given an actual operational rating of S_o1, since the impact is indiscernible, but still presents a weakness. Potentially of course, the rating is much higher since compromising the pairing could lead to any number of attacks (including against

```
+-----+---------------------------------+--------------+--------------------------------------------+
| No. | Attribute                       | Rating (S0-S4)| Evidence                                  |
+-----+---------------------------------+--------------+--------------------------------------------+
| ACT | Address: xx:xx:xx:34:8A:2D      | Sp2,So0      | Address unique to each device              |
| POT |                                 | Sp3,So0      | Vehicle Tracking possible                  |
| ACT | OUI: xxxxxxxxxxxxxxxxxxxxxxxxx  | Sp1,So0      | Anonymous data acquired                    |
| POT |                                 | Sp1,So3      | Further recon may lead to So3              |
| ACT | OS: NULLOS                      | Sp0,So0      | No data acquired                           |
| POT |                                 | Sp1,So3      | Further recon may lead to So3              |
| ACT | Generic services (no suggestive) | Sp1,So0     | Anonymous data acquired                    |
| POT |                                 | Sp1,So0      |                                            |
| ACT | Generic services (no bespoke)   | Sp1,So0      | Anonymous data acquired                    |
| POT |                                 | Sp1,So0      |                                            |
| ACT | Service port no.: 7             | Sp1,So1      | Sync services found                        |
| POT |                                 | Sp3,So3      | Unauthorised sync --> Sp3, Fuzzing --> So3 |
| ACT | Generic services found (no PAN) | Sp1,So0      | Anonymous data acquired                    |
| POT |                                 | Sp1,So0      |                                            |
| ACT | Pairing: legacy                 | Sp1,So1      | Legacy pairing found,                      |
|     |                                 |              | susceptible to MITM, deprecated spec       |
| POT |                                 | Sp3,So2      | Attack goal/services offered dependent     |
|     |                                 |              | --> data loss, MITM or DoS                 |
| ACT | Open RFCOMM ports: [1, 4]       | Sp1,So1      | Port scan invisible, anonymous data acquired |
| POT |                                 | Sp3,So3      | Open ports --> Sp3, port scan --> DoS, So3 |
| ACT | Open L2CAP ports: [1, 3, 23, 25] | Sp1,So1     | Port scan invisible, anonymous data acquired |
| POT |                                 | Sp3,So3      | Open ports --> Sp3, port scan --> DoS, So3 |
| ACT | Flooding: indiscernible impact  | Sp1,So0      | System information, no impact              |
| POT |                                 | Sp1,So3      | More testing could cause further DoS       |
| ACT | ObexStress: no impact,          | Sp1,So0      | System information acquired, no impact     |
|     |             system info acquired |             |                                            |
| POT |                                 | Sp1,So3      | Further testing could cause further DoS    |
+-----+---------------------------------+--------------+--------------------------------------------+
```

Figure 12: Severity classification ratings for the denial of service suite

privacy).

The presence of open ports meant that an actual operational rating of S_o1 was assigned, since this allowed us to acquire system information. However, operationally, the right `AT` commands or number of packets to send could be enumerated and this may result in a higher potential rating. Since there was no OBEX FTP service on the vehicle, both the mounting and directory traversal attacks were not carried out, and therefore both were automatically assigned S_p0, S_o0 on all fronts.

The first sets of results from the denial of service tests in Figure 12 are identical to the data extraction results since the reconnaissance aspects were performed in the same vehicle. They are included in the figure for clarity. For the purposes of the denial of service tests, only the last two set of results are discussed.

Flooding caused no discernible impact, however, we were able to enumerate the maximum transmission units of the open L2CAP ports on the vehicle, therefore system information was available. This resulted in the assignment of S_p1 (for anonymous data acquired), but since there was no discernible impact, the operational rating was set at S_o0. Potentially, however, further testing (with more specific unit sizes, or with more packets or over a longer period of time) could cause a denial of service in any number of Bluetooth functionalities in the worst case scenario.

Likewise, stressing open RFCOMM ports caused no impact, although again system information was required (even if it was the fact that the action was forbidden) and therefore given the same rating as above. Potentially, finding the right combination and length of malformed data could also cause denial of service, and the ratings adjusted accordingly.

5.2. Aftermarket devices

Following the same process as testing through the native headunit connection (Section 4), the severity classification was created based on findings and observations (recall that S_p is the privacy severity rating, and S_o the operational severity rating). The table given in Figure 13 gives an overview of the severity classification for each of the results in conjunction with manual observation for tests performed with aftermarket devices.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	CAN data present	Sp1,So0	Anonymous CAN data returned
POT		Sp4,So3	Profiling on CAN possible, reverse engineering, exploits could cause damage
ACT	VIN no. found	Sp2,So0	VIN number allows vehicle identification
POT		Sp4,So0	Multiple vehicle tracking possible
ACT	Caused significant reaction	Sp1,So3	Anonymous data, significant impairment
POT		Sp1,So4	Simultaneous connections to devices attached in many cars could cause multiple vehicle impairment

Figure 13: Severity classification from tests with aftermarket devices

As can be seen from the results, no personal data was acquired. This was to be expected since we are interfacing with the CAN bus and its connected ECUs, rather than the infotainment system where any personal data is most likely to be stored. Only system information was acquired, which is the reason for the S_p1 ratings. This may not hold true in a worst case scenario as CAN traffic can be used to profile individual drivers through usage patterns [40]. Therefore the potential privacy rating was set at S_p4.

The VIN number was also acquired, which allows for the identification of the individual vehicle, as well as individual characteristics including make, model, year of registration, airbag type and more. With this, vehicle tracking may also be possible, as there are many online tools such as determining tax status that makes use of this number.

Finally, the last characteristic deals with whether there was significant operational effect on the vehicle. As could be seen from the case study as well as from the systematic evaluation, almost anything is possible on the vehicle should the correct CAN message be determined. Reverse engineering the right CAN message (including any diagnostic messages) allows for significant operational impairment of the vehicle, hence the potential indicator of S_o3.

Potentially, many of these devices (being small and the OBD-II port hidden from general view) could be planted, and a signal sent to every device within range. This led to the assignment of the potential indicator of S_p1 (for anonymous data acquired) and S_o4 (for possible significant impairment on multiple vehicles - even simultaneously should they all be in range at once).

Like the classification before, prioritisation could take place depending on the results. The edge cases of S_p0,S_o0 and any S4 rating could be dealt with straightaway. The former could be used as evidence in a security assurance case that there is low risk associated with that aspect of the component, or as evidence that there are sufficient countermeasures in place. The latter (as it affects multiple vehicles severely) would be a priority in any case.

The middle cases (S2 and S3 ratings and combinations thereof) would require other factors weighting it to form priorities (see Section 6). Since we are testing the internal CAN network here, operational factors might be given precedence, as typically there is very little personal data available on the CAN bus.

Recall that validation of a black box with unknown inputs and no set of expected outputs is very difficult. The black box nature also means that formal verification is unfeasible due to lack of knowledge of internal behaviours. Therefore validation of this classification was also performed through domain expert review.

6. Discussion and Conclusions

Results from the classification process can be used to inform further development immediately.

The assigned security levels can be used to prioritize the development of countermeasures. Anything that is classified as S_p0,S_o0 can be added to the security assurance case as low risk, and therefore low priority. Conversely, anything with a classification of S4 in any aspect can be targeted for the development of countermeasures since this rating indicates a risk to multiple vehicles. When the development of these countermeasures is considered complete, comparisons can be drawn with the initial rating, and if considered acceptably addressed, a description of these countermeasures can also be added to the security assurance case as evidence of risk analysis and risk reduction.

Deciding precedence between equal but different classifications, such as S_p1,S_o2 and S_p2,S_o1 would depend, in part, on the components being targeted. These tests are performed on the headunit where personal data is most likely to be stored, so privacy might potentially be given more precedence. For example, given the choice between S_p1,S_o2 and S_p2,S_o1, the latter might be the more likely candidate to target for improvement. Conversely, the aftermarket device is directly connected to the CAN network on which vehicle operations are highly dependent, but personal data is highly unlikely to be found. Therefore the operational rating would take precedence although this may be apparent anyway as no identifying data would mean the privacy rating would only ever remain at S_p1.

The above may seem intuitive given only two parameters, but in the event where all four aspects of the EVITA classification scheme are in play, this could aid in the decision making process.

Finally, the rationale given for the worst-case scenario (or the potential ratings) is intended to be a guideline to the starting point for either new test cases based on the information acquired during the reconnaissance phase. Alternatively, these can be used as guidance to what might be done by a malicious adversary to complete the attack, since many of the tests pull up just short of an invasive attack due to the risk to test vehicles.

6.1. Contribution

Our contribution is the process we present in this paper to classify results (Section 3.1.3 and Section 3.1.4) of a systematic security evaluation using severity ratings.

This process could be integrated into the appropriate vulnerability testing and penetration testing activities recommended by J3061 during the development phases at the system, hardware and software levels. Each of the actual (ACT) ratings (resulting from the actual tests) can be considered absolute ratings, and could be augmented with additional risk assessment factors such as cost of attack, ease of attack, opportunity available and knowledge of the attacker at the discretion of the manufacturer. These additional factors were considered out of scope within this paper.

The potential (POT) ratings could be treated in a similar way. Since they represent worst case scenarios, they could feature in the threat analysis and risk assessment activities recommended by J3061 during the concept phase, and again, additional risk weighting factors could be defined by the manufacturer. This ensures flexibility to deal with different implementations and concepts.

Once these classifications take place, they could be used as evidence in security assurance cases not dissimilar to the proposed Automotive Security Assurance Levels (ASEALs) [3] or the cybersecurity integrity levels proposed in [45]. Placing it in such a structured manner could also help scope the breadth and depth of tests to be performed, in accordance with the priority of the testers or the owners of the systems-under-test.

6.2. Conclusion

In conclusion, the systematic Bluetooth evaluation carried out enumerated manufacturer-specific implementation details regarding both the information and entertainment system as well as through the diagnostics port. We were able to affect the vehicle by injecting both OBD-II specific and raw CAN messages.

Once testing was completed, severity ratings that were assigned could be used to prioritise development of countermeasures and to add evidence to a security assurance case. The rationale behind the worst-case scenario ratings could be used as guidance for further tests. A severity classification for these results was validated using domain expert review. Finally the wider issues of Bluetooth discoverability, and what manufacturers could consider to counter these dangers was briefly explored.

Future work includes extending the proposed test result classification to include both the safety and financial aspects of the severity classification. An extension of the methodology by including additional risk factors such as attack cost or difficulty would also mean that a more granular classification could be achieved. The question of how a feedback loop may be formed to the early stage threat analysis and risk assessment should also be addressed. This question highlights the need to consider security by design, and investigations into the precise points in the design and implementation process this could be added would be desirable.

7. Acknowledgements

The authors would like to acknowledge Anthony Jude (HORIBA MIRA) for his input as a domain expert reviewer, and to Alastair Ruddle (HORIBA MIRA) for valuable comments. Thanks also go to Olivier Haas (Coventry University) for his support during this research.

References

[1] R. Charette, This Car Runs on Code, IEEE Spectrum 46 (3).

[2] A. Pretschner, M. Broy, I. H. Kruger, T. Stauner, Software engineering for automotive systems: A roadmap, in: Proc. of the 2007 Future of Software Engineering, IEEE Computer Societies, Minneapolis, MN, 2007, pp. 55–71.

[3] S. Bayer, T. Enderle, D. K. Oka, M. Wolf, Security Crash Test Practical Security Evaluations of Automotive Onboard IT Components, in: Electronic Proceedings of the 2014 ESCRYPT Automotive Safety and Security Conference, ETAS, Stuttgart, 2015.

[4] E. Santos, D. Schoop, A. Simpson, Formal Models for Automotive Systems and Vehicular Networks : Benefits and Challenges, in: O. Altintas, E. Ekici, M. Tsai, M. Sepulcre, B. Bloessl, Y.-L. Wei (Eds.), Proceedings of the 2016 IEEE Vehicular Networking Conference (VNC), no. 1029, IEEE, Columbus,OH, 2016.

[5] S. Gürgens, C. Rudolph, A. Mana, S. Nadjm-Tehrani, Security Engineering for Embedded Systems - the SecFutur vision, in: B. Hamid, C. Rudolph, C. Rulan (Eds.), Proceedings of the 2010 International Workshop on Security and Dependability for Resource Constrained Embedded Systems, ACM, ACM New York, NY, Vienna, Austria, 2010, p. 7.

[6] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, S. Ravi, Security as a new dimension in embedded system design, in: Proceedings of the 41st Design Automation Conference (DAC), ACM, San Diego, CA, 2004, p. 753.

[7] S. Ravi, A. Raghunathan, P. Kocher, S. Hattagandy, Security in Embedded Systems: Design Challenges, ACM Transactions on Embedded Computer Systems 3 (3) (2004) 461–491.

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, Comprehensive Experimental Analyses of Automotive Attack Surfaces., in: Proceedings of 20th USENIX Security Symposium, USENIX Association, San Francisco, CA, 2011, pp. 77–92.

[9] A. Bertolino, Software Testing Research : Achievements , Challenges , Dreams, in: Proceedings of the 2007 Conference on Future of Software Engineering (FOSE), IEEE Computer Society, Minneapolis, 2007, pp. 85–103.

[10] M. Cheah, S. Shaikh, J. Bryans, H. N. Nguyen, Combining third party components securely in automotive manufacture, in: Proceedings of 10th International Conference on Information Security Theory and Practice, Springer, Crete, 2016.

[11] M. Cheah, S. A. Shaikh, O. Haas, A. Ruddle, Towards a systematic security evaluation of the automotive Bluetooth interface, Journal of Vehicular Communications 9 (2017) (2017) 8–18.

[12] C. Miller, C. Valasek, Remote Exploitation of an Unaltered Passenger Vehicle, Tech. rep., Las Vegas, NV (2015).
URL `http://illmatics.com/RemoteCarHacking.pdf`

[13] R. Verdult, F. Garcia, J. Balasch, Gone in 360 seconds: Hijacking with Hitag2, in: Proceedings of 21st USENIX conference on Security symposium, USENIX Association Berkeley, Bellevue, WA, 2012, p. 37.

[14] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, Experimental Security Analysis of A Modern Automobile, in: Proceedings of the 2010 IEEE Symposium on Security and Privacy, IEEE, Oakland, CA, 2010, pp. 447–462.

[15] T. Hoppe, S. Kiltz, J. Dittmann, Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats, in: Proceedings of the 28th International Conference on Computer Safety, Reliability, and Security (SAFECOMP), Springer-Verlag Berlin, Hamburg, 2009, pp. 145–158.

[16] Argus, Argus Cyber Security Working With Bosch to Promote Public Safety and Mitigate Car Hacking (2017).
URL `https://argus-sec.com/argus-cyber-security-working-bosch-promote-public-safety-mitigate-car-hacking`

[17] S. Woo, H. J. Jo, D. H. Lee, A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN, IEEE Transactions on Intelligent Transport Systems 16 (2) (2015) 993–1006.

[18] Y. Liu, I. Traore, Systematic security analysis for service-oriented software architectures, in: Proceedings of ICEBE 2007: IEEE International Conference on e-Business Engineering - Workshops: SOAIC 2007; SOSE 2007; SOKM 2007, 2007, pp. 612–621.

[19] I. Schieferdecker, J. Grossmann, M. Schneider, Model-Based Security Testing, in: Electronic Proceedings in Theoretical Computer Science (EPTCS 80), Vol. 80, Tallinn, Estonia, 2012, pp. 1–12.

[20] N. Madenas, A. Tiwari, C. Turner, S. Peachey, An analysis of supply chain issues relating to information flow during the automotive product development, Journal of Manufacturing Technology Management 26 (8) (2015) 1158–1176.

[21] M. S. Idrees, Y. Roudier, L. Apvrille, A framework towards the efficient identification and modeling of security requirements, in: Proceedings of the 5th conference on Network Architecture and Information Systems, Menton, 2010, pp. 1–15.

[22] A. van Lamsweerde, Elaborating Security Requirements by Construction of Intentional Anti-Models, in: Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), IEEE Computer Society, Edinburgh, 2004, p. 10.

[23] D. Xu, K. E. Nygard, Threat-Driven Modeling and Verification of Secure

Software Using Aspect-Oriented Petri Nets, IEEE Transactions on Software Engineering 32 (4) (2006) 265–278.

[24] A. Fuchs, S. Gürgens, G. Pedroza, L. Apvrille, EVITA Project - Deliverable D3.4.4: On-Board Architecture and Protocols Attack Analysis, Tech. rep., Fraunhofer SIT (2008).
URL `http://www.evita-project.org/Deliverables/EVITAD3.4.4.pdf`

[25] Research Institutes of Sweden, HEAVENS (2017).
URL `https://www.sp.se/en/index/research/dependable_systems/heavens/sidor/default.aspx`

[26] Microsoft Developer Network, The STRIDE Threat Model (2005).
URL `https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx`

[27] M. M. Islam, A. Lautenbach, C. Sandberg, T. Olovsson, A Risk Assessment Framework for Automotive Embedded Systems, Proc. of ACM International Workshop on Cyber-Physical System Security (2016) 3–14.

[28] SAE International, J3061 : Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (2016).
URL `http://standards.sae.org/j3061_201601/`

[29] Horiba MIRA Ltd., ISO26262: Update on development of standard (2016).
URL `https://nmi.org.uk/wp-content/uploads/2016/01/HORIBA-MIRA-ISO-26262-NMI-Jan-16-DDW.pdf`

[30] P. J. Brooke, R. F. Paige, Fault trees for security system design and analysis, Computers & Security 22 (3) (2003) 256–264. doi:10.1016/B978-0-7506-6843-9.00021-4.

[31] S. Mauw, M. Oostdijk, Foundations of Attack Trees, in: D. H. Won, S. Kim (Eds.), Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC 2005), no. im, Springer Berlin Heidelberg, Seoul, South Korea, 2005, pp. 186–198.

[32] Penetration Testing Execution Standard, PTES Technical Guidelines (2014).
URL http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

[33] IEEE, IEEE Standards Association: Registration Authority (2017).
URL https://regauth.standards.ieee.org/standards-ra-web/pub/view.html\#registries

[34] J. P. Dunning, Taming the blue beast: A survey of bluetooth based threats, IEEE Security and Privacy 8 (2) (2010) 20–27.

[35] J. Dunning, Spooftooph (2012).
URL http://tools.kali.org/wireless-attacks/spooftooph

[36] R. Chang, V. Shmatikov, Formal analysis of authentication in bluetooth device pairing, in: Proceedings of the 2007 Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA), Wroclaw, Poland, 2007, p. 45.

[37] A. Levi, E. Çetintas, M. Aydos, Ç. K. Koç, M. U. Çaglayan, Relay Attacks on Bluetooth Authentication and Solutions, in: Proceedings of the 19th International Symposium of Computer and Information Sciences (ISCIS'04), Springer-Verlag Berlin Heidelberg, Kemer-Antalya, 2004, pp. 278–288.

[38] K. Haataja, Security threats and countermeasures in Bluetooth-enabled systems, Phd thesis, University of Kuopio (2009).
URL http://epublications.uef.fi/pub/urn_isbn_978-951-27-0111-7/urn_isbn_978-951-27-0111-7.pdf

[39] K. Haataja, K. Hypponen, Man-in-the-middle attacks on Bluetooth: a comparative analysis, a novel attack, and countermeasures, in: Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing, IEEE, St. Julians, 2008, pp. 12–14.

[40] M. Enev, A. Takakuwa, K. Koscher, T. Kohno, Automobile Driver Fingerprinting, Proceedings on Privacy Enhancing Technologies 2016 (1) (2016) 34–50.

[41] Robert Bosch GmbH, CAN Specification version 2.0, Tech. rep., ROBERT BOSCH GmbH, Stuttgart, Germany (1991).
URL `http://esd.cs.ucr.edu/webres/can20.pdf`

[42] SAE International, SAE J1979 E/E Diagnostic Test Modes (2014).
URL `http://standards.sae.org/j1979_201408/`

[43] ISO, ISO 14229-1: Road Vehicles - Unified Diagnostic Services, Standard ISO/TC 22/SC 31 (2013).

[44] Sparkfun Electronics, ELM327 AT Commands (2010).
URL `https://www.sparkfun.com/datasheets/Widgets/ELM327_AT_Commands.pdf`

[45] D. Ward, P. Wooderson, Automotive cyber-security integrity levels, in: Proceedings of the 11th International Conference on System Safety and Cyber-Security (SSCS 2016), IET, London, 2016, pp. 10—19.