# Towards a systematic security evaluation of the automotive Bluetooth interface

Cheah, M, Shaikh, SA, Haas, O & Ruddle, A

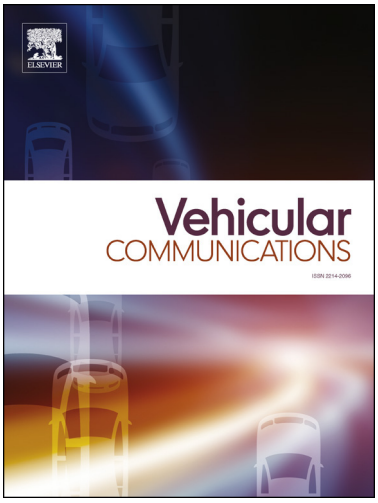


**Original citation & hyperlink:**
Cheah, M, Shaikh, SA, Haas, O & Ruddle, A 2017, 'Towards a systematic security evaluation of the automotive Bluetooth interface' *Vehicular Communications*, vol 9, pp. 8-18
http://dx.doi.org/10.1016/j.vehcom.2017.02.008

DOI 10.1016/j.vehcom.2017.02.008
ISSN 2214-2096

Publisher: Elsevier

# Accepted Manuscript




Towards a systematic security evaluation of the automotive Bluetooth interface

Madeline Cheah, Siraj A. Shaikh, Olivier Haas, Alastair Ruddle

PII: S2214-2096(16)30147-4
DOI: http://dx.doi.org/10.1016/j.vehcom.2017.02.008
Reference: VEHCOM 84

To appear in: *Vehicular Communications*

Received date: 19 October 2016
Revised date: 22 December 2016
Accepted date: 23 February 2017

Please cite this article in press as: M. Cheah et al., Towards a systematic security evaluation of the automotive Bluetooth interface, *Veh. Commun.* (2017), http://dx.doi.org/10.1016/j.vehcom.2017.02.008

# Towards a systematic security evaluation of the automotive Bluetooth interface

Madeline Cheah[a,*], Siraj A. Shaikh[a], Olivier Haas[a], Alastair Ruddle[b]

[a] *Centre for Mobility and Transport Research, Coventry University, Coventry, UK*
[b] *Future Transport Technologies, HORIBA MIRA, Nuneaton, UK*

**Abstract**

The modern vehicle requires connectivity in order to enable and enhance comfort and convenience features so desired by customers. This connectivity however also allows the possibility that an external attacker may compromise the security (and therefore the safety) of the vehicle. In order to answer this problem, we propose a framework for a systematic method of security testing for automotive Bluetooth interfaces and implement a proof-of-concept tool to carry out testing on vehicles using this framework. From our findings, we conclude that the method enabled us to enumerate multiple weaknesses and that by continuing to extend the work, we would discover more.

*Keywords:* automotive security, operational and field testing, threat model, wireless security, Bluetooth

## 1. Introduction

The modern vehicular system is opening up, with wireless interfaces and services implemented for customer comfort and convenience. The introduction of these interfaces means that malicious external influences are now possible, as demonstrated by seminal experimental analyses on individual vehicles [11, 29, 38]. These influences can be construed as "cyberattacks" or "hacks", which have come to mean an attempt to manipulate an insecure system to cause negative consequences such as harm, damage or destruction. In cyber-physical systems - defined as a system where computational and physical processes are integrated [30] - the harm may not be limited to logical assets (such as personal data theft or loss) but could conceivably also cause physical harm, such as is the case with a vehicle. Protection and defence mechanisms are therefore necessary in order to mitigate or nullify the consequences of an attack. Several challenges stand in the way of implementation although experimental analyses on a vehicle's possible entry points have been performed. The primary concern here

*Corresponding author
*Email addresses:* cheahh2@uni.coventry.ac.uk (Madeline Cheah), siraj.shaikh@coventry.ac.uk (Siraj A. Shaikh), o.haas@coventry.ac.uk (Olivier Haas), alastair.ruddle@horiba-mira.com (Alastair Ruddle)

is that the placement and details of countermeasures requires knowledge as to where, in the system, security vulnerabilities or weaknesses exist in the first place, and what its nature is.

Bluetooth is a pervasive interface and was therefore chosen for this study because of the potential negative impact should it be compromised. There have been estimates that vehicles with a Bluetooth interface number at nearly nine million currently, with a forecast of 21 million vehicles to have Bluetooth by 2018 [19]. Market growth for information systems, of which Bluetooth is a major enabler, is anticipated to grow to $1.6 billion by 2020, with at least a 40% rise in automotive wireless technology [2]. Bluetooth is a low power, short range wireless communication technology, capable of forming ad-hoc networks. Security issues with this technology are well documented [15].

The main contribution of the paper is a systematic method of evaluating the security of the automotive Bluetooth interface, something that has not yet been performed. This is needed to maximise the effectiveness of the security evaluation and is implemented through a proof-of-concept tool based on attack tree modelling and penetration testing methods. This tool was then used to evaluate the Bluetooth interface on a range of built-in automotive infotainment systems.

The rest of this paper is structured as follows: Section 2 discusses related work, whilst Section 3 looks at Bluetooth security, both generally and specific to the automotive domain. We describe our methodology in Section 4 and present our proof-of-concept tool development and validation in Section 5. We discuss our findings in Section 6 and consider future directions in Section 7.

## 2. Related work

There are several challenges with securing wireless interfaces in vehicles. 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, 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. Even post-release, patches, unless performed over-the-air, for discovered vulnerabilities are difficult to apply once units are sold.

All of the above is dependent on acquiring knowledge and information regarding existing vulnerabilities and holds true not just of Bluetooth attacks, but also generally. Some exploits have already been demonstrated in literature on the vehicle as a whole [11, 29] or on various subsystems [22, 42, 46, 51, 52], some are reported through "hacker" conferences such as Black Hat [38] whilst still others can be inferred through technological trends.

Although these papers show an impressive range of experimentation and an in-depth knowledge of the target system, they have not mapped out a process or taxonomised their findings. Furthermore, information on the practical aspect of security testing is scarce; because automotive systems are complex with many different technologies integrated into the single vehicle, many papers dealing with experimental analysis by necessity limit their

scope to a single interface, protocol or technology which are extremely diverse in nature. Of the papers that involve practical security analysis on vehicles, only one details attacks on an automotive system (at a high level) via Bluetooth [11], although many agree that Bluetooth is a viable entry point for an attacker [42, 56, 14, 36, 22, 25]. Despite the paucity of information, from the number and variety of reported threats, vulnerabilities and exploits, it is clear that a systematic description of the problem is required.

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 - which by implication means a subjective prioritisation of what and where to test [32] - potentially results in flaws being overlooked. A methodical approach increases the likelihood of determining flaws, thereby mitigating this problem [48]. Systematic analyses can also be supported by a variety of tools and utilities, for example, through the use of graph-based modelling, and in this case also means that, not only is the final result documented, but all the details that led to the system compromise [13].

Systematic evaluations have been described in model-based testing studies such as [34] and security specific model-based testing [48] is an active field of research. These have inspired our method of systematism, in particular the use of attack trees. However, although this approach provides rigour and confidence, we have no trustworthy model from which to generate tests. This is because the Bluetooth specification is embedded in other systems (such as the embedded system's operating system and other firmware) for which we would need to include to provide a complete model representation of the implementation and for which there is very little information. Furthermore, whilst model-based security testing may provide coverage of security weaknesses in a system, applications thereof (e.g. [23]) have required that models be available or pre-built in order to formally examine. The barrier to using such methods is as above, that the information required to do so is not available, both due to commercial confidentiality and the obscurity of subcomponents within the system (many of which are third party). This also procludes other methods of enabling systematic evaluation such as attack graphs, for which formal model checking could be performed.

Automotive specific systematic methods of evaluation are described in the "E-safety vehicle intrusion protected applications" (EVITA) project [17]. The EVITA project ultimately aims to provide a secure architecture for automotive on-board networks and evaluates the realisation of this using two "views" the first of which is a magnified view. Attack tree modelling (discussed further in Section 4.2) is used to support these processes, although the end goal of verifying whether assets are really protected somewhat differs from the aim of this paper which is to identify unprotected assets through a methodical evaluation. The second view, called a compositional view, deals with looking at attack categories (and related security guarantees) to ensure that omitted attacks are minimised. The latter is a valuable exercise, however, where a system already exists with unknown properties (and therefore unknown guarantees) as is the case with this paper, the ability to analyse coverage in such a way is limited. Methodical evaluation methods are also presented in the J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [47], drawing from EVITA, although

information provided has been examples thereof rather than application to a system.

## 3. Bluetooth

Bluetooth is more complex than most wireless standards, due in part to the Frequency Hopping Spread Spectrum (FHSS) mechanism designed to reduce narrowband interference. Channel hopping occurs once every 625 $\mu$s and in some cases also uses Adaptive Frequency Hopping (AFH), whereby channels that can cause interference are avoided [8]. Data whitening is also performed by XOR-ing each packet with a pseudorandom sequence, in order to facilitate signal transmission.

Adding to the complexity is also the fact that not all Bluetooth implementations are identical; Bluetooth standards specify various service profiles that could be used in order to customise the technology, whether that be to enable "hands-free" communication, allow file transfers or grant access to phonebooks and messages [4]. Profiles consist of information regarding dependencies, user interface details and specific protocols required by the service. This information is vital in detailing what the device is capable of doing, and, from an adversary's point of view, also gives information on potential weaknesses. The vast majority of services embodied by these profiles communicate via the Radio Frequency Communications (RFCOMM) and Logical Link Control and Adaptation Protocol (L2CAP) layers and, where there is an open channel, could be used to send or extract data. The number and nature of accessible ports on a remote device depend on the services being offered along with whether a user is paired and connected.

The pairing process, essentially the method by which two or more devices synchronise their "hops", is well documented and in the interest of brevity is only outlined here. A complete introduction may be found in [8]. The pairing process uses one of two mechanisms:

- **Legacy pairing:** This has been superseded by Simple Secure Pairing (SSP) in the Bluetooth 2.1 specification, although many older platforms still use this mechanism. The pairing exchange involves the derivation of a link key from the Bluetooth address, the PIN and a random number. This link key is then stored locally and used in subsequent authentication and encryption processes. The primary danger to this mechanism is the fact that the PIN is the only aspect providing entropy, exacerbated by the fact that PINs often contain only four decimal digits.

- **SSP:** There are four association models under the SSP umbrella, these being i) "Out-of-band" (using non-traditional channels to complete the pairing process), ii) Numeric comparison (where two devices with screen capabilities both output a number which the user then confirms as identical), iii) Passkey entry (where one device displays a PIN, which is then keyed into another device) and iv) Just Works (where both devices have no input or output capabilities and pairing takes place without any further authentication.

### *3.1. Bluetooth vulnerabilities*

Table 1: Bluetooth attack classification (adapted from [15])

| *Attack classification* | *Threats* |
|---|---|
| Surveillance | Includes general scans (or war-nibbling), inquiry scans and brute scans to determine non-discoverable addresses. Manufacturers can be profiled using organisationally unique address bits. Also includes service enumeration. |
| Range extension | Most consumer devices are Class 2, with a range of up to 10 metres. Range can be extended through the use of external directional antenna or passive radio locators. |
| Obfuscation | Includes spoofing or cloning a device name, class, address or service profile fingerprint. Can serve to further other actions such as man-in-the-middle attacks. |
| Fuzzing | Injection of arbitrary or malformed data. |
| Sniffing | Using Bluetooth narrowband or wideband receivers or tools in order to dump raw data from a connected Bluetooth interface. |
| Denial of service (DoS) | Flooding with data, or jamming signals to cause applications or devices to freeze or crash or battery exhaustion. |
| Malware | Infection from malicious programs via Bluetooth interface. |
| Unauthorised direct data access | Includes targeting hard-coded default PINs, brute forcing PINs, targetting vulnerable implementations of APIs, sending commands via covert channels to extract data, or using loopholes in the object exchange (OBEX) protocols. |
| Man in the middle (MITM) | Masquerading as a trustworthy entity, or injecting oneself in the middle of a communication in order to eavesdrop on or modify data, as described by [20]. |

There are many categories of attacks that could be performed. These are summarised in Table 1. The trends abstracted from the specific techniques play an important part in identifying the ultimate goals of a potential attacker. These trends can be distilled further into the categories of data extraction, data manipulation and denial of service. The data extraction goal, as a proof-of-concept, forms the basis for the attack tree presented in this paper.

### *3.2. Automotive Bluetooth*

The wireless nature of Bluetooth has been attractive to automotive manufacturers as a way of reducing weight and wiring in the vehicle, along with the hands-free services that Bluetooth can offer. The latter is driven in large part by the advent of regulations barring the use of mobile phones in vehicles. Its flexibility means that manufacturers can offer customised features to end users.

Bluetooth implementation on vehicles differs from conventional PC and mobile platforms. The software on vehicles may not have been updated in years and older chips are in use even in newer vehicles, with many still using legacy pairing. Presented information is customised by device and not by users (so no distinction is made between users of the same remote device) with user information potentially centrally held [45]. Although it has been posited that requiring user interaction within the authentication process increases security [57] (for example with numeric comparison), many vehicles use other pairing mechanisms such as passkey entry (often with a default universal static PIN [15]). The front-end of the system may not ask for user confirmation or display alerts (such as when an unauthorised device is trying to pair) as might be expected in other embedded systems.

Additionally, a vehicle is mobile and is rarely stationary with the ignition turned on. This, combined with a relatively short range of ten meters could pose a challenge to an attacker. Range extension (see Table 1) however has been used successfully to extend the range to about a mile, inject audio and eavesdrop on in-cabin conversations [41]. Furthermore, compromise could also occur pre-travel (for example in a carpark or a garage) for possible disruption later on.

The majority of built-in infotainment systems either search for a device to pair with or require a user to actively enable Bluetooth [42], though the seeming security of the latter is diminished given that not every vehicle limits the time in which the interface is discoverable. Additionally, many implementations look for previously paired devices and may initiate a connection without switching on the discoverable mode; potential attackers could also test for the existence of a device via a name inquiry. An adversary could then wait for the opportune moment once the existence of a device is known to pair and form a connection with the target.

## 4. Methodology

Automotive security is a diverse field, with full functional specifications unlikely to be readily available due to commercial sensitivity. Combined with the fact that there is little work to build on (see Section 2), the lack of information necessitates a black box approach.

The methods used in this paper are empirical, and derived from standard practice in the security industry, namely threat analysis using attack trees, with penetration testing methods employed to populate these trees. The premise of these methods is founded on testing a system from an attacker's point of view to identify system weaknesses and to reflect what an adversary might face in reality.

### *4.1. Threat modelling*

Threat modelling is the process by which security threats can be determined, analysed and documented [33]. A threat can be defined as any potential harmful event that could compromise an asset (an object of value). Combinations of attack vectors and methods are usually employed in order to realise these threats.

This process typically follows the process of identifying a threat (synonymous in this case with an attacker goal), which can be broken down into sub-goals iteratively until individual actions are identified [33]. Many threat models can broadly be taken to represent the decision making process of a potential adversary. Popular methods include Microsoft's STRIDE (a mnemonic for the threat categories of spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege), DREAD (damage potential, reproducibility, exploitability, affected users and discoverability) and visualisation tools such as Data Flow Diagrams (DFDs) [33, 31] all of which help to classify, assess the risk of and visualise the threat landscape. These methods can be used in combination with constructions such as attack trees in order to further enumerate the threat [26], although in these cases, vulnerabilities have already been identified in an emulated environment where the full system is known, which is not the case in this paper. The attack tree (which might be instead subsumed under a different category of vulnerability tree [28]) was used in order to explore exact paths to the pre-discovered vulnerability [26].

### *4.2. Attack trees*

Many structures exist in order to model security-related testing processes. Examples include attack nets, which are customised Petri nets with places representing states or modes of interest, and transitions that represent events such as input or commands [37]. Although eminently suited to singular activities, such as bringing together seemingly unconnected flaws to form an individual attack path, representing relationships between different attacks (especially on poorly documented systems) is more challenging [37].

Attack trees were first developed to describe the security of systems [49] in a structured manner and are conceptual diagrams meant to illustrate threats from an attacker's point of view. These trees can be represented diagrammatically (Figure 1) or textually (Figure 2). Attack trees focus on abuse cases (in this case an attack), and even in an informal capacity, can support threat assessment. This information would usually need to be further formalised, empiricised or investigated (if resources and available data permits), but is nevertheless a useful starting point for threat identification [43].

Attack trees can be considered analogous to the more common concept of fault trees. The primary difference between the two structures can largely be attributed to paradigm. Where a fault tree looks at random faults that could cause an undesired event, the attack tree concentrates on intentional malicious actions that could cause the system to enter an undesired state [6].

Like fault trees, intermediate events (or a branch that can be further developed with leaf nodes) are connected by logic gates such as AND and OR [49], and where temporal order is necessary SEQUENTIAL AND (SAND). Where the AND logic gate is used, an attack (a parent node) is considered complete only when all the steps (child nodes) are completed and

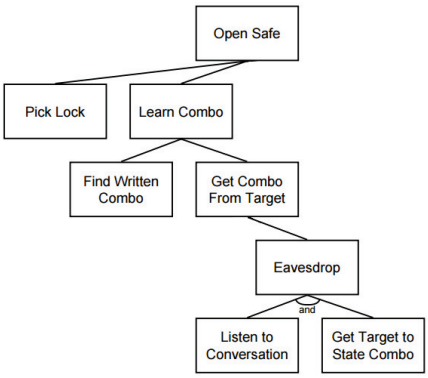



Figure 1: An example attack tree detailing how to open a safe [49]

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

Figure 2: Textual representation of an attack tree detailing how to open a safe [49]

where necessary in sequence (when SAND is used). With an OR logic gate, achievement of any one of the steps is adequate to complete the attack. Leaf nodes can be assigned boolean (such as possible or impossible) or continuous (such as cost) values.

The structure is acyclic, requiring a root (attack goal), and is directional, which is significant. From a design perspective, a top-down approach, where an attack goal is first identified followed by all subsequent methods of achieving the goal, early in the development life-cycle, is recommended [50]. From a testing perspective, however, this is challenging because of the black box nature of security testing. Since the system already exists for us, the tree here is built bottom-up tracing from leaf to root, based on observable entry points and subsequent behaviours when probed, leading to potential attack goals: the very process that penetration testing is based on. Note that although the structure is acyclic, the process of security testing as presented here (requiring multiple iterative test runs) can be considered cyclic.

The sample tree shown in Figure 3 has been developed iteratively. As there are no real-world measures for detection of security incidents on a vehicle, the primary method of validation remains domain expert input (such as is the case for building the tree from bottom up [55]) and data from practical applications; this best practice has been used by others [7].

### *4.3. Penetration testing*

This form of testing is heavily dependent on the flaw hypothesis approach [37], defined as a method of identifying deviations from functional specifications [53]. Testing is then performed to discover capabilities that can be "exploited to violate some aspect of security [policies]" [53]. Methods used are usually not prescribed, although some, such as identifying machine addresses, are more common than others. The ultimate aim is to force the system into either entering a disallowed state, or executing a disallowed action, thereby exposing the weakness that allowed such an action.

The result of every action taken in penetration testing can be considered deterministic based on system implementation, configuration and state [21]. This implies a very large

(possibly infinite) number of results based on any number of implementation-configuration-state combinations. The number of combinations also means that test coverage is not easily quantified. However, evidence compiled through the testing process can be used to provide a security assurance case. This is not without its limitations:

- Firstly, proving the complete absence of insecurities in an implementation is not possible, as tests only ever expose a limited subset of vulnerabilities [18, 16]. It can only be stated that under certain abuse cases, these flaws were not present and that this is acceptable security;

- Secondly, a method that does manage to end in exploitation may not be the only method that does, however, the underlying flaw is exposed and can be addressed for that particular method. Abstracted patterns of this method can also be extracted (for example, a buffer overflow exploit is mechanically the same whatever the system) to test for similar weaknesses through other vectors.

- Finally, there are concerns with generalisation of a system since testing tends to rely on implementation. However, as automotive production lines are standardised, we reasonably assume that if the Bluetooth stack in a vehicle is flawed in some way, this same vulnerability may appear in some other vehicle of same age, make, model and software version (of which there may be millions). Furthermore, reuse is a common approach to reducing cost in the industry [44], and as such flaws could be replicated even in newer models. Even vulnerabilities that have been patched in more modern embedded systems may be present in newer vehicles, as software in an automotive system is updated less frequently [40].

The preliminary round of data gathering (results of which are in Section 5) from all test systems was conducted using the Penetration Testing Execution Standard (PTES), which is not a formal standard but rather a set of guidelines to provide an outline of the process. Formal technical standards do exist [1], however, the primary barrier to adopting them is the high information needs about the testing environment [27].

Recall that results are deterministic based on implementation, configuration and state. Despite this, results can neither be predicted nor calculated when a black box perspective is employed. Therefore all experimentation is empirical. Nonetheless, there is value in carrying out such research. The list of vulnerabilities may be unenumerable, but corrective action to address an observed vulnerability would reduce that list by one, whilst also providing the owner of the system with information for improvements and verification of current configurations.

## 5. Proof-of-concept tool

The proof-of-concept tool created to carry out the steps as detailed in the sample attack tree is an extension of the concepts embodied by various other proof-of-concept, pre-alpha and beta Bluetooth security testing tools created since 2003. Some examples include *redfang*

[54] (a proof-of-concept tool created in 2003 for brute-scanning), *CarWhisperer* [41] (a small tool created in 2005 to scan for manufacturers that implement hardcoded PINs and use that to connect to and inject audio into or record audio from a vehicle), or *Bluesnarfer* [35] (a tool created to exploit a vulnerability discovered in 2003, using AT commands in order to extract phonebooks from susceptible mobile phones). The most recent release of *nOBEX* [39] can be construed as one of the most relevant as it is directed at automotive headunits. However, functionality is currently limited to fuzzing and they make no claims as to automation.

The tool itself depends on the official Linux Bluetooth stack (called *Bluez* [5]) to provide the Bluetooth functionality needed to test these interfaces. It was developed using Python 2.7 on a Kali Linux system using the the Bluetooth Python extension module *Pybluez* [24]. The tree structure and the tree search facility is enabled by the *treelib* library [12].

Table 2: Bluetooth tool verification (identifying information has been redacted)

| *Device* | *Characteristics* |
|---|---|
| 1 | Bluetooth address: XX:XX:XX:93:94:07<br>Bluetooth version 4.0 on Android 5.1, service profiles obtained<br><br>Filesystem is mountable and browsable, file transfer possible assuming pairing and connection, responds to AT commands assuming user gives permission on the appropriate channels. |
| 2 | Bluetooth address: XX:XX:XX:40:41:47<br>Bluetooth version 2.1 on BlackberryOS 7.1, service profiles obtained<br><br>No OBEX File Transfer Profile (FTP) for mounting or file transfer. Responds to OBEX object push commands and AT commands assuming user gives permission on the appropriate channels |
| 3 | Bluetooth address: XX:XX:XX:4A:19:80<br>Bluetooth version 4.0 on Windows Phone 8.0, service profiles obtained<br><br>No OBEX FTP. Responds to OBEX object push commands and AT commands assuming user gives permission on the appropriate channels |

Many of the pre-built tools found were singular in nature, essentially providing only a single aspect of testing (such as spoofing, perhaps leading to MITM). Furthermore, the majority are now archived or unsupported and dependent on deprecated libraries. This necessitated the creation of a new tool incorporating many different functions in order to facilitate systematism.

The proof-of-concept tool presented in this paper follows a predefined attack tree in order to complete a penetration test. Note that some aspects of the attack tree are difficult to carry out practically (as is the case with actions involving social engineering), and there are some assumptions in order to facilitate development; that the Bluetooth systems are discoverable (as brute scanning is impractical) and that a connection is possible. The tool

during development stages was then tested on a variety of multiple older mobile platforms to verify functionality (Table 2).

The predefined sample tree (Figure 3) concentrates on data extraction as an attack goal and is used as an input into the tool constructed (Algorithm 1). This process travels down each node of the tree, until it reaches the leaves, and depending on the logic gate, carries out the necessary testing steps, recording and outputting data to the appropriate test run.

**Algorithm 1:** Data extraction via the Bluetooth interface. Vehicle data refers to any data that is available from the vehicle, including personal data, vehicle-generated data, or data about the vehicle itself.

```
input  : Predefined attack tree
output: Vehicle data
initialization;
for AttackGoal do
    foreach AttackTreeBranch in order do
        foreach AttackTreeLeaf do
            if AttackTreeLeaf is OR then
                while attack fails do
                    AttackSteps on vehicle;
                end
                Record vehicle data;
                if no vehicle data then
                    Display AttackSteps and AttackTreeLeaf;
                else
                    Populate AttackTreeLeaf with vehicle data;
                end
            else
                perform all AttackSteps;
                Record vehicle data;
                if no vehicle data then
                    Display parent nodes with children for AttackSteps;
                else
                    Populate AttackTreeLeaf with vehicle data;
                end
            end
        end
    end
    for AttackTreeLeaf do
        if has vehicle data then
            for AttackTreeLeaf do
                Display AttackTreeBranch;
            end
        else
            for Empty AttackTreeLeaf do
                Display AttackSteps;
            end
        end
    end
end
```

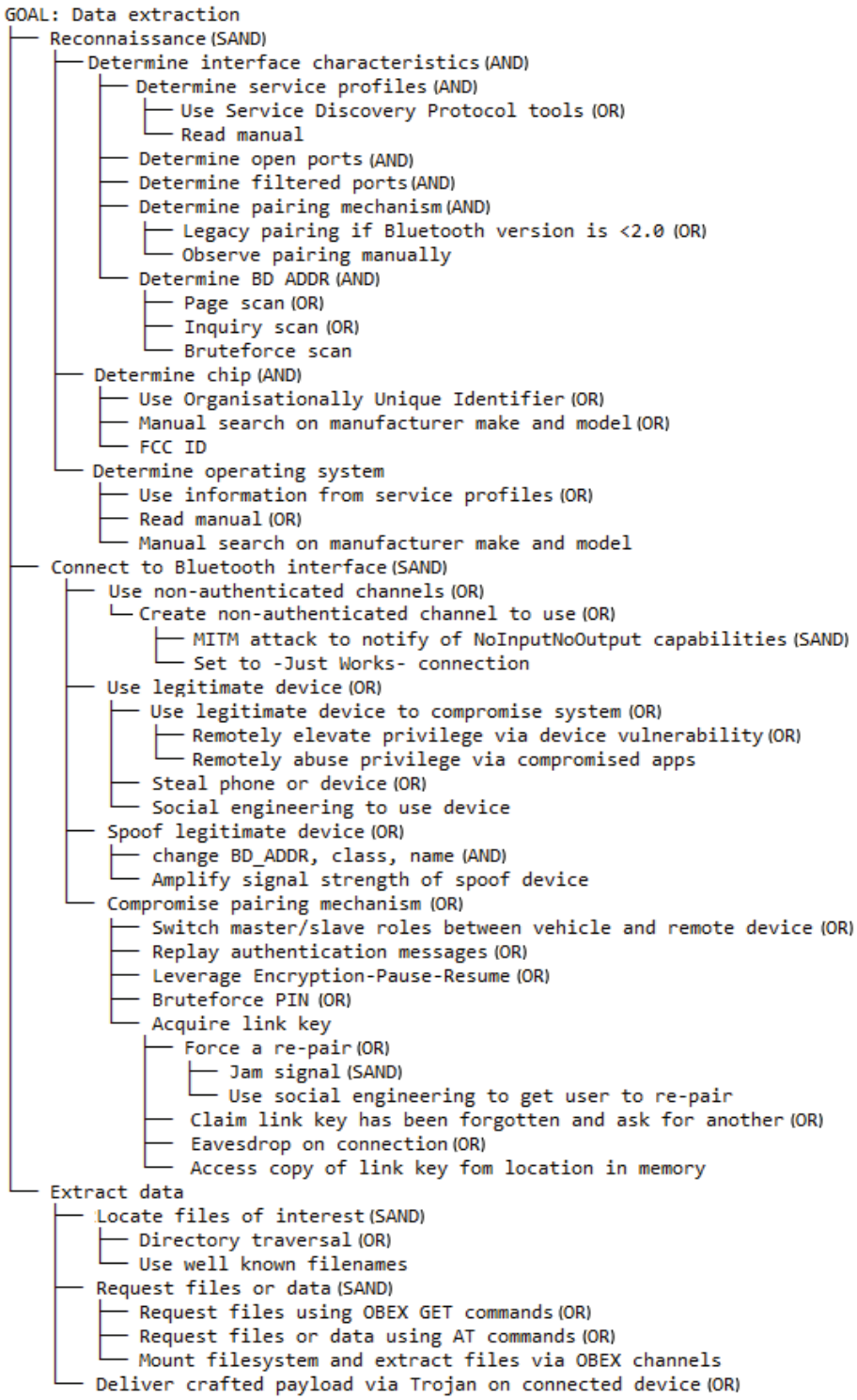



Figure 3: Textual representation of a sample attack tree based on data extraction as an attacker goal

### *5.1. Automation*

The features of the tool (Table 3) can be categorised broadly into:

- **Reconnaissance**, which can be defined as a survey of the system's existence, configuration and capabilities,
- **Connection attributes**, which includes information on pairing mechanisms, transmission sizes and connection state, and
- **Attack goal (data extraction)**, which encompasses methods that would allow the realisation of the attack goal

The tool is semi-automated (Table 3), in that many aspects of the test suite does not require manual intervention. This is true of the majority of the reconnaissance and connection attribute determination features where the tool will run down the tree on its own should all data be available. If data is not available for a particular branch, the steps (i.e. the subtree) that would need to be completed would be displayed for possible manual intervention.

Some manual decision making is required when performing the data extraction tests. Whilst the individual attack methods (such as sending in AT commands) can be automated, full automation of multiple types of attacks in sequence is difficult in this case as the target system is a black box, and the search for weaknesses in such an environment comes with a large number of sequential decision making issues [21]. This is in addition to uncertainties around what could be construed as an attacker's "point-of-view" (the paradigm of penetration testing) and interconnection between different attack components [21] when testing a system.

A key consideration during conception and development of the tool was the choice of nodes to automate. There were some aspects where non-trivial development was required, when it would have been simpler to observe the target device (such as which of four SSP mechanisms is in use during pairing), and others where manual intervention was required as the information is generally held physically. An example of the latter is the device's Federal Communications Commission (FCC) ID, which provides valuable information on the attributes of wireless communications, and is usually printed on a physical label.

In summary, the tool, whilst being only semi-automated, provides a head start with regards to establishing the security state baseline for the target system. Furthermore, the attack tree methodology underlying the tool also provides for a traceable and systematic set of results.

### *5.2. Tool validation*

The tool was tested against five different vehicles (of different makes, models and ages), all of which were registered within the last five years. There was no additional source of information regarding the Bluetooth implementation on these systems, other than what was publicly available through the owner's manuals. Due to commercial sensitivity, identifying information has been redacted. Results of tests on vehicles are summarised in Table 4.

Table 3: Tool Features (corresponds to *AttackSteps* in Algorithm 1)

| | *Feature* | *Automation stage* |
|---|---|---|
| *Reconnaissance* | Discovery of 'discoverable' device addresses using inquiry scans | Automated |
| | Discovery of 'hidden' devices using brute-force scanning | Automated |
| | Determination of Bluetooth chip manufacturer using the organisationally unique ID (OUI) to scan through a database of stored OUIs | Automated |
| | Determination of service profiles offered by device using the Service Discovery Protocol (SDP) | Automated |
| | Preliminary indication of device operating system (OS) using indicators in discovered service profiles | Semi-automated (may require manual search from other sources) |
| | Determination of whether device uses legacy pairing by checking the Bluetooth version | Semi-automated (a Bluetooth version of 2.0 or below indicates legacy pairing) |
| | Determination of open ports by sending information to all possible RFCOMM and L2CAP ports and awaiting responses | Automated |
| | Determination of filtered ports by sending information to all RFCOMM and L2CAP pors and filtering for specific error messages | Automated |
| *Connection Attributes* | Determination of pairing status with reference to local paired devices | Automated |
| | Pair or unpair the device as appropriate with reference to local paired devices | Semi-automated (the user decides to pair or unpair) |
| | Checks for presence of OBEX File Transfer Profile (FTP) and OBEX Object Push Profile (OPP) service with reference to discovered service profiles | Automated |
| | Checks maximum transmission unit (MTU) for open L2CAP ports by sending increasing size of packets until Bluetooth error 90 ('message too long') appears | Automated |
| *Data Extraction* | Attempted extraction of information using modem attention (AT) commands through open RFCOMM ports | Semi-automated (open ports are automatically determined and AT commands can be sent in, but user chooses the commands). |
| | Attempted extraction of information by mounting and browsing the filesystem on a FUSE based filesystem type | Semi-automated (checks for OBEXFTP profile automatically, tries to mount automatically, but user does the manual browsing if successful) |
| | Attempted extraction of information using OBEX GET and PUT commands | Semi-automated (checks for open RFCOMM ports automatically, user is given choice of commands) |
| *Outputs* | Scan logs written to .csv or .txt files and collated at the end of test run | Automated |
| | Populated attack tree displayed and logged | Automated |
| | Subtrees displayed and logged where test results have not been found or entered. This is denoted by `NULL`. Appropriate subtrees are found using width-first search. | Automated |

Table 4: Practical testing on built-in infotainment systems in test vehicles

| *Interface characteristics* | *Outcome* |
|---|---|
| **BT address, version and class**<br>XX:XX:XX:34:8A:2D<br>Bluetooth v. 2.0 , 0x340408 (AV, Hands-free)<br><br>**Pairing mechanism**<br>Legacy: vehicle produces dynamic 6 digit PIN<br><br>**Services**<br>HFP, SyncML Server, A2DP, AVRCP, PBAP (Client), OBEX OPP, MAP MNS<br><br>**Open ports (when paired)**<br>RFCOMM 1, 4 and L2CAP 1, 3 | **User feedback**<br>Audio and visual notice of successful pairing, device added to paired list. User is not alerted to any of the attempted actions beyond pairing<br><br>**Actions and results**<br>Responds to AT commands on RFCOMM channel 4 with "AT+BRSF=39", vehicle ignores commands on all other channels (no response from vehicle), ignores OBEX PUT, OBEX GET and SyncML client. Filesystem cannot be mounted |
| **BT address, version and class**<br>XX:XX:XX:6E:DC:D5<br>Bluetooth v. 2.0 , 0x340408 (AV, Hands-free)<br><br>**Pairing mechanism**<br>Legacy: user chooses number of digits as PIN<br><br>**Services**<br>SPP, OBEX OPP, OBEX FTP, IrMC Sync, HFP, HSG, PANU | **User feedback**<br>Visual change informing user that PIN should be chosen, no alert that pairing was successful or that there were attempted connections. Remote device not added to paired devices list<br><br>**Actions and results**<br>Could pair but not connect; port scan could not be performed. All actions beyond reconnaissance ended in error |
| **BT address, version and class**<br>XX:XX:XX:CF:69:B8<br>Bluetooth v 2.1 , 0x340408 (AV, Hands-free)<br><br>**Pairing mechanism**<br>SSP - Numeric comparison, however vehicle reported pairing with default PIN of '0000' with SSP disabled on test adaptor.<br><br>**Services**<br>HFP, AVRCP, A2DP, PBAP (Client), MAP MNS, OBEX OPP<br><br>**Open ports (when paired)**<br>RFCOMM 1,2,3,4,5 and L2CAP 1 | **User feedback**<br>Numeric comparison proceeded as normal. Pairing using 0000 created a very brief visual alert, device added to paired devices list.<br><br>No other alert was issued for subsequent actions performed on vehicle<br><br>**Actions and results**<br>Bluetooth error 104 (connection reset by peer) on all open channels for AT commands issued. Ignores OBEX PUT and GET. Filesystem cannot be mounted |

| *Interface characteristics (continued)* | *Outcome (continued)* |
|---|---|
| **BT address, version and class**<br>XX:XX:XX:8A:81:20<br>Bluetooth v. 2.0 , 0x300408 (AV, Hands-free)<br><br>**Pairing mechanism**<br>Legacy: user chooses number of digits as PIN<br><br>**Services**<br>PBAP (Client), AVRCP, OBEX OPP, SyncML Server, HFP<br><br>**Open ports (when paired)**<br>RFCOMM 1, 4 and L2CAP 1, 23 | **User feedback**<br>Visual change informing user that PIN should be chosen, visual alert informing of successful pairing. Remote device added to paired devices list.<br><br>**Actions and results**<br>Only an unstable connection to vehicle Bluetooth system given. Vehicle responded to SyncML client on test device requesting for contact synchronisation and established a session. However, the session ended before any information came through as connection terminated. AT commands were ignored (no response from vehicle) on open channels. |
| **BT address, version and class**<br>XX:XX:XX:C3:4A:64<br>Bluetooth v2.0 , 0x340408 (AV, Hands-free)<br><br>**Pairing mechanism**<br>Legacy: 4 digit PIN, default 0000<br><br>**Services**<br>HSG, Sync, HFP, OBEX OPP, Update, Vendor specific SPP1, Vendor specific SPP2, PBAP (Client), A2DP, AVRCP, OBEX FTP, PANU<br><br>**Open ports (when paired)**<br>RFCOMM 3,4,5,8,9,10,11,12 and L2CAP 1, 15, 25, 27 | **User feedback**<br>Small visual alert in secondary screen above steering wheel, no alert from main screen. No flags or alerts for all subsequent actions. Device added to paired devices list.<br><br>**Actions and results**<br>AT commands are ignored (no response from vehicle), except on channel 5 where no information is returned. Filesystem is mountable on a FUSE based filesystem and browsable. Two folders were found: “recorder” and “update_ftp”. It is unclear what the purpose and permissions of these folders are. No response to OBEX GET or PUT. |

## 6. Discussion and conclusions

To effectively enumerate the security state of the automotive Bluetooth interface, we used the structured form of the attack tree to carry out a penetration test, based on data extraction as a goal.

### *6.1. Pairing and connection*

Four out of the five vehicles used the legacy pairing mechanism exclusively. The third vehicle tested used SSP (Numeric Comparison), however, disabling SSP on the local test

laptop meant that pairing could occur using '0000'. This confirms that, even with relatively modern vehicles, older technologies are still in use and could therefore be vulnerable to eavesdropping and MITM attacks. Even where the newer SSP pairing mechanism is in use, bypassing such measures were straightforward because of the weakness of the default PIN. The number of open ports is usually dependent on whether a user is paired or connected to the vehicle, however, the OBEX OPP profile remained open regardless. Although data extraction through this port was unsuccessful, techniques such as repeated pushing of files through this port could help fulfill alternative attack goals (such as denial of service).

Vehicles universally reconnected with the test laptop as soon as it came into range, the ramification of which is that an attacker would only need to compromise the pairing process once. The pairing process differed (at least mechanically) on the vehicles tested, with some generating PINs, others with hardcoded PINs, and still others asking the users to select the PIN. The window in which a vehicle remains discoverable also varied. Out of the five, three vehicles had a two minute window, which limits attacker opportunity. The other two held the discoverable window open indefinitely, which leaves the vehicle open to opportunistic adversaries.

### *6.2. User feedback*

All of these vehicles do not alert the user as to any actions taken during an active connection. Thus, AT requests, filesystem mounting, synchronisation commands, OBEX PUSH and GET commands all took place without any visual or aural warning; the limitation of finding a Bluetooth address aside, should the user be unaware that a pairing or connection had taken place, an adversary could carry out many of these attacks unnoticed. Additionally, although at this point in time data extraction was the focus, there were open ports even where a pairing had not taken place. The danger in this is the ability to flood these ports with data thereby, potentially, causing a denial of service. Although this was not performed as the focus was on data extraction, the details of this would certainly be included in any future attack tree involving denial of service as an attack goal.

### *6.3. System weaknesses*

An interesting finding was the ability to mount a filesystem with full read and partial write access (in that directories could be written, but not files); through this entry point, any number of crafted applications could be placed on to the vehicle to disrupt operations. Fuzzing here (such as directory names containing non-standard characters) could also reveal more about this feature.

There were also vendor specific profiles found on the last of the vehicles in Table 4. They appear to be serial port profiles, although its functionality would need to be probed further. AT commands elicited Bluetooth error 104 (connection reset by peer), but future work may include traffic sniffing and analysis during normal course of operations which might yield more information.

Since hands-free phone calls are one of the more prominent features of Bluetooth in vehicles, it was unsurprising to see the Phone Book Access Profile (PBAP) on some of the vehicles. This allows phonebooks to be synchronised from a mobile phone (or other device

holding a phonebook in the correct format) to the vehicle. The 'client' status denotes that this only goes one way, from the remote device to the vehicle, however, tools such as *nOBEX* could be used to fuzz this particular feature by uploading contacts or phone numbers that have non-standard characters, or are past a certain length.

Another feature of interest was that a synchronisation profile (SyncML Server, Sync, IrMC Sync) was present in all but one of these vehicles. These synchronisation profiles are generally used to synchronise phonebooks and other personal information between phone and vehicle. Although there was no personal data extracted, in at least one instance, a connection (albeit unstable) with a SyncML client was established; the setup could be revised to try and correct for this and verify whether any data could be extracted.

The last point of interest was the presence of the Personal Ad-Hoc Network User (PANU) profile on two of the vehicles. This service is able to transfer Ethernet packets across a connection. There are three security modes used by this profile. The first is "non-secure", where a device does not initiate any security procedures. The second is service-level enforced security, where security procedures are not initiated before a channel is established at L2CAP level. Lastly, the link-level enforced security mode initiates security procedures before the link set-up at the Link Management Protocol (LMP) layer [3]. LMP controls the radio link between two devices. The mode used by the PANU profile in this case is so far unenumerated, but represents a potential alternative method to send in (Ethernet) packets that could compromise a vehicle.

### *6.4. Limitations*

Despite the fact that the class of device was set to that of a smartphone with telephony features (0x5a020c or 0x7a020c), the vehicles universally recognised the test laptop as a media player rather than a phone. This meant that certain features (such as phonebook) were unavailable when interacting with the vehicular user interface. However, the version of *Bluez* (5.37) used in this paper does not support the Bluetooth hands-free profile (HFP), which may explain the discrepancy, and future work would require repeating the tests on a downgraded version in order to verify this. This may also be the reason behind being unable to connect to one of the vehicles.

Five vehicles is also a small number to test, but the case could be built further by testing a wider range and larger number in order to help further verify and validate what has already been acquired.

It should be noted also that quantitative metrics at this point in time is outside of scope, as detection measures of intrusion within a vehicle are at a nascent stage at best, and "known" vulnerabilities are generally based on inference from other applications of Bluetooth. Additionally, practical testing is concentrated on built-in systems as aftermarket "car kits" differ in that they would not usually have connection to the in-vehicle network and, in this piece of work, could be considered a lower priority at this point in time.

### *6.5. Conclusions*

From our findings, we conclude that there are multiple weaknesses in a vehicle that could be exploited, and that consideration should be given by vehicle manufacturers as to

how legacy pairing could be replaced, as well as how visible interactions with the vehicle should be. We also conclude that these possible vulnerabilities were found by following the structured procedure presented in this paper, and we were able to test them to build a security case. Whilst no personal data was found in the exploration of these vehicles, there was data from the vehicle itself. Additionally, the method is extensible with the addition of other attack goals, and could be used to test other potential weaknesses such as open ports when unpaired. Along with the above, we have also gained some insight into and established a security state baseline of the implementation and configuration of Bluetooth within the vehicle. This allows us to edge towards a stage where we could build a model for more rigorous tests, such as model-based testing.

Mitigating the vulnerability found in one branch may also cancel out other branches where attack paths are not so easily tested. For example, mounting an operating system could be performed using a legitimate device, or by acquiring a phone through social engineering - negating the ability to mount an operating system would close off both pathways. We envision that this tool would help manufacturers both in evaluating current implementations and using the results to help secure future iterations of designs.

## 7. Future work

The sample tree presented only represents one of the three categories of attack goals planned, with the other two being denial of service and data manipulation. At the heart of the framework is the initial reconnaissance of any test interface, and which forms the basis for all future vulnerability hypotheses and threat models. Although there is a myriad of tools available, the information they provide collectively is scattered at best, and a method of bringing these together in a coherent format would be desirable. To further this, we would extend and create new attack trees to include other goals, along with tests on a wider range of vehicles, with multiple versions of the Linux Bluetooth stack in order to correct for any setup issues.

Looking further into the future, an in-depth analysis of the operating system would also help define how this module interacts with the backbone Controller Area Network (CAN) bus that controls vehicle operations. The work so far (which only looks at the entry point and the Bluetooth feature) will thus form a stepping stone into research on possible disruptions on vehicle operations from an external source.

We aspire to use the work here to harden future systems by generating and incorporating security specific requirements from the results of the systematic security evaluation into future designs [10]. We also envision that further testing would provide more insight into how components behave, which would enable us to start the process of building a trustworthy model that could be used in more rigorous processes such as model based security testing.

Wireless connectivity is only increasing, both in the number and variety of interfaces and protocols used as well as bandwidth capacities. Going forward to where vehicles become “smarter” and highly automated [9], the need to evaluate the security of wireless connections will never be less than essential.

## Acknowledgements

Thanks go to Paul Wooderson (HORIBA MIRA) and Jeremy Bryans (Coventry University) for valuable comments.

## References

[1] Bayer, S., Enderle, T., Oka, D. K., & Wolf, M. (2015). Security Crash Test – Practical Security Evaluations of Automotive Onboard IT Components. In *ESCRYPT Automotive Safety and Security*. Stuttgart.

[2] Bluetooth SIG (2013). Bluetooth SIG Analyst Digest H2 2013 (cached).

[3] Bluetooth SIG Inc. (2003). Personal Area Network (PAN).

[4] Bluetooth SIG Inc. (2015). Bluetooth profiles overview.

[5] BlueZ Project (2015). BlueZ.

[6] Brooke, P. J., & Paige, R. F. (2003). Fault trees for security system design and analysis. *Computers & Security*, *22*, 256–264.

[7] Byres, E., Franz, M., & Miller, D. (2004). The use of attack trees in assessing vulnerabilities in SCADA systems. In *Proceedings of the 2004 International Infrastructure Survivability Workshop (IISW'04), IEEE*. Lisbon: IEEE.

[8] Chai, E., Deardorff, B., & Wu, C. (2012). Hacking Bluetooth.

[9] Cheah, M., & Shaikh, S. (2015). Autonomous Vehicle Security. *IET Engineering and Technology Reference*, *1*.

[10] Cheah, M., Shaikh, S. A., Bryans, J., & Nguyen, H. N. (2016). Combining third party components securely in automotive systems. In S. Foresti, & J. Lopez (Eds.), *Proceedings of the Information Security Theory and Practice: 10th IFIP WG 11.2 International Conference, WISTP 2016, Heraklion, Crete, Greece, September 26–27* (pp. 262–269). Springer International Publishing.

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

[12] Chen, X. (). Treelib documentation (revision bd53bbdf).

[13] Dawkins, J., & Hale, J. (2004). A systematic approach to multi-stage network attack analysis. *Proceedings - Second IEEE Information Assurance Workshop*, (pp. 48–56).

[14] Department for Transport (2015). The Pathway to Driverless Cars : A detailed review of regulations for automated vehicle technologies.

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

[16] Felderer, M., Zech, P., Breu, R., Buchler, M., & Pretschner, A. (2015). Model-based security testing: a taxonomy and systematic classification. *Software Testing Verification and Reliability*, *26*, 119–148.

[17] Fuchs, A., Gürgens, S., Pedroza, G., & Apvrille, L. (2008). *EVITA Project - Deliverable D3.4.4: On-Board Architecture and Protocols Attack Analysis*. Technical Report Fraunhofer SIT.

[18] Geer, D., & Harthorne, J. (2002). Penetration testing: a duet. In *Proceedings of the 18th Annual Computer Security Applications Conference, 2002* (pp. 185–195). Las Vegas, NV: IEEE Computer Society.

[19] GSMA (2013). *Connected Car Forecast: Global Connected Car Market to Grow Threefold within Five Years*. Technical Report GSMA.

[20] Haataja, K., & Hypponen, K. (2008). Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures. In *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on* March (pp. 12–14). St. Julians: IEEE.

[21] Hoffman, J. (2015). Simulated Penetration Testing : From " Dijkstra " to " Turing Test ++ ". In *Proceedings of the 25th International Conference on Automated Planning and Scheduling*. Jerusalem, Israel.

[22] Hoppe, T., Kiltz, S., & Dittmann, J. (2011). Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures. *Reliability Engineering & System Safety*, *96*, 11–25.

[23] Jürjens, J. (2008). Model-based Security Testing Using UMLsec. A Case Study. *Electronic Notes in Theoretical Computer Science*, *220*, 93–104.

[24] karulis (). Pybluez: Bluetooth Python extension module.

[25] Kim, Y., & Kim, I. (2013). Security Issues in Vehicular Networks. In *Proceedings of the 2013 International Conference on Information Networking (ICOIN)* (pp. 468–472). Bangkok, Thailand: IEEE.

[26] Klöti, R., Kotronis, V., & Smith, P. (2013). OpenFlow : A Security Analysis. In *Proceedings of the 21st IEEE International Conference on Network Protocols*. Göttingen.

[27] Knowles, W., Baron, A., & Mcgarr, T. (2015). *Analysis and recommendations for standardization in penetration testing and vulnerability assessment*. Technical Report Lancaster University London.

[28] Kordy, B., Pièvre-Cambacédès, L., & Schweitzer, P. (2014). DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, *13-14*, 1–38.

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

[30] Lee, E. A. (2008). Cyber Physical Systems : Design Challenges University of California , Berkeley. In *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)* (pp. 363–369). Orlando, FL: IEEE.

[31] Liu, B., Shi, L., Cai, Z., & M.Li (2012). Software Vulnerability Discovery Techniques: A Survey. In *Proceedings of the 4th International Conference on Multimedia Information Networking and Security*. Nanjing, China: IEEE.

[32] Liu, Y., & Traore, I. (2007). 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* (pp. 612–621).

[33] Marback, A., Do, H., He, K., Kondamarri, S., & Xu, D. (2012). A threat model-based approach to security testing. *Software - Practice and Experience*, *43*, 241–258.

[34] Marinescu, R., Saadatmand, M., Bucaioni, A., Seceleanu, C., & Pettersson, P. (2014). A Model-Based Testing Framework for Automotive Embedded Systems. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 38–47). IEEE.

[35] Martelloni, R. (2012). Bluesnarfer.

[36] Mason, S. (2012). Vehicle remote keyless entry systems and engine immobilisers: Do not believe the insurer that this technology is perfect. *Computer Law & Security Review*, *28*, 195–200.

[37] Mcdermott, J. P. (2000). Attack Net Penetration Testing. In M. E. Zurko, & S. J. Greenwald (Eds.), *Proceedings of the 2000 Workshop on New Security Paradigms* (pp. 15–21). Cork, Ireland: ACM New York, NY.

[38] Miller, C., & Valasek, C. (2015). *Remote Exploitation of an Unaltered Passenger Vehicle*. Technical Report Las Vegas, NV.

[39] NCC Group (). nOBEX.

[40] Nilsson, D. K., & Larson, U. E. (2008). Secure Firmware Updates over the Air in Intelligent Vehicles. In *ICC Workshops - 2008 IEEE International Conference on Communications Workshops* (pp. 380–384). IEEE.

[41] n—u: The Open Security Community (2012). Carwhisperer, Bluetooth Attack.

[42] Oka, D. K., Furue, T., Langenhop, L., & Nishimura, T. (2014). Survey of Vehicle IoT Bluetooth Devices. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications* (pp. 260–264). Matsue, Japan: IEEE.

[43] Opdahl, A. L., & Sindre, G. (2009). Experimental comparison of attack trees and misuse cases for

security threat identification. *Information and Software Technology*, *51*, 916–932.

[44] Pretschner, A., Broy, M., Kruger, I. H., & Stauner, T. (2007). Software engineering for automotive systems: A roadmap. In *Proceedings of the FOSE'07 2007 Future of Software Engineering* (pp. 55–71). Minneapolis, MN: IEEE Computer Societies.

[45] QNX (). QNX Software Development Platform: Managing User Accounts.

[46] Rouf, I., Miller, R., Mustafa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., Seskar, I., Ishtiaq, R., & Sangho, O. (2010). Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium* (p. 21). Washington, DC: USENIX Association Berkeley.

[47] SAE International (2016). J3061 : Cybersecurity Guidebook for Cyber-Physical Vehicle Systems.

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

[49] Schneier, B. (1999). Attack Trees: Modeling Security Threats.

[50] Torr, P. (2005). Demystifying the threat modeling process. *IEEE Security & Privacy Magazine*, *3*, 66–70.

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

[52] Verdult, R., Garcia, F. D., & Ege, B. (2013). Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer. In *Supplement to the Proceedings of the 22nd USENIX Security Symposium*. Washington, DC: USENIX Association.

[53] Weissman, C. (1995). *Penetration Testing (Handbook for the Computer Security Certifications of Trusted Systems)*. Technical Report Naval Research Laboratory Washington, DC.

[54] Whitehouse, O., Halsall, S., & Kapp, S. (). redfang.

[55] Wolf, M., Scheibel, M., & Gmbh, T. U. V. I. (2012). A Systematic Approach to a Quantified Security Risk Analysis for Vehicular IT Systems. In *ESCRYPT Automotive Safety and Security* (pp. 195–210). Karlsruhe: ESCRYPT.

[56] Wolf, M., Weimerskirch, A., & Paar, C. (2004). Security in Automotive Bus Systems. In *Proceedings of the Workshop on Embedded Security in Cars (ESCAR)* (pp. 1–13).

[57] Yee, K. (2002). *User interaction design for secure systems*. Technical Report May University of California Berkeley, California.