Combining Third Party Components Securely in Automotive Systems

Madeline Cheah(✉), Siraj A. Shaikh, Jeremy Bryans, and Hoang Nga Nguyen

Centre for Mobility and Transport Research, Coventry University, Coventry, UK
cheahh2@uni.coventry.ac.uk,
{siraj.shaikh,jeremy.bryans,hoang.nguyen}@coventry.ac.uk

Abstract. Vehicle manufacturers routinely integrate third-party components and combining them securely into a larger system is a challenge, particularly when accurate specifications are not available. In this paper, we propose a methodology for users to introduce or strengthen security of these composed systems without requiring full knowledge of commercially sensitive sub-components. This methodology is supported by attack trees, which allow for systematic enumeration of black box components, the results of which are then incorporated into further design processes. We apply the methodology to a Bluetooth-enabled automotive infotainment unit, and find a legitimate Bluetooth feature that contributes to the insecurity of a system. Furthermore, we recommend a variety of follow-on processes to further strengthen the security of the system through the next iteration of design.

Keywords: Automotive security · Attack trees · Secure design · Security testing · Bluetooth

1 Introduction

Automotive security has become an issue with the advent of smarter vehicles, which incorporate a large variety of external facing interfaces that could be used to maliciously affect vehicles. The context of our work is the way in which various components are combined to achieve the final vehicle product. Components are often generic with many general purpose features. This promotes their reuse, which drives overall costs within the supply chain down. Larger components are often provided as whole "off-the-shelf" subsystems (for example an infotainment unit), with each component originating with a different manufacturer. Within the automotive supply chain, system integrators often do not have the final detailed designs of the components, especially where these components represent intellectual property such as source code. Components for which no privileged information is available are often referred to as "black boxes" [11], with "white boxes" being those for which all information is available. This distinction becomes important when testing the integrated system [13].

S. Foresti and J. Lopez (Eds.): WISTP 2016, LNCS 9895, pp. 262–269, 2016.
DOI: 10.1007/978-3-319-45931-8_18

The contribution of this paper is a methodology for the secure combination of third party components. The methodology includes a systematic and semi-automated penetration testing process supported by attack trees. This leads to the identification of additional security requirements over and above the functional and integration requirements that already exist for the system, which can then be used to improve the design of the system with respect to security. The motive for beginning the process with testing is to acquire confidence with regard to the overall implementation. The testing process moves knowledge of the component along the black-white spectrum, where we can then extract requirements for secure behaviour in the given context to help mitigate security flaws. This is particularly valuable where a system contains many third party components of which even the original equipment manufacturer (OEM) may not have complete sight because of commercial sensitivities.

The remainder of this paper is structured as follows: we review related work in Sect. 2, followed by an outline of our proposed methodology in Sect. 3. We then apply this methodology to a case study in Sect. 4. We discuss the implications thereof and our conclusions in Sect. 5.

2 Related Work

There are comparative approaches to each of the stages of our methodology, and as such our survey has been divided into categories of gathering security requirements, threat assessment and attack trees, along with a brief discussion on the automotive specific cybersecurity standard J3061.

Security Requirements. Similar methods for gathering security requirements have been proposed by [6], in that security requirements are linked to possible attacks. A key difference to our methodology however is that a functional model of the system is required, which is more information than is usually available in a black box system. Attack trees in a requirements gathering and actioning process are also used in the System Quality Requirements Engineering (SQUARE) methodology [7]. However, use cases in this methodology concentrated on application to a company's procedures rather than embedded systems.

Threat Assessment. This process determines threats (defined as potential negative events that could compromise an asset) to the surface of the target system, typically by looking at the potential malicious actions. In the automotive domain, empirical studies have already shown that attacks on vehicular components are possible [3]. However, despite impressive experimental analyses, actions taken to compromise the vehicle and their results were not systematised. This, in addition to the "grey box" nature of automotive components led us to penetration testing for threat assessment, supported by attack trees, in order to determine the initial security state of the system relative to the target attack goal.

Attack Trees. Attack trees are diagrams modelling the potential actions of an attacker to reach an attack goal [19]. They have been discussed as a possible threat modelling technique in the automotive specific SAE cybersecurity standard J3061 [18], which draws from the "E-safety vehicle intrusion protected applications" (EVITA) project. It is for this reason that we have chosen to use this method. Furthermore, attack trees can help inform threat assessment even in an informal capacity [16]. Formal methods such as attack graphs are not feasible as there is not enough up-front information about the target system.

These trees can be represented diagrammatically or textually. Logic gates (AND and OR) are also commonly used within these trees. Where AND is used, an attack (parent node) is considered complete only if all steps have been completed. Where OR is used, the parent node is complete if at least one of the steps is completed. These gates are also sometimes referred to as conjunctive and disjunctive refinements respectively [12]. For application purposes, where temporal order may be a concern, sequential AND (SAND) could also be used.

A related approach is the formation of "anti-models" [10], depicting how model elements may be threatened (analogous to attack trees). However, these anti-models are derived from the model of the system-to-be (with attendant high informational needs), which makes it less suitable for a black-box system. Even where there are methods that allow for only partial specifications (such as the framework based on Model Driven Engineering) [9], perfectly legitimate behaviour in those specifications could actually be a weakness in terms of the larger system boundary.

EVITA and J3061. EVITA elaborates on some of the possible usages of attack trees. Deliverable 2.3 also includes an outline in which security requirements could be traced back to the attack tree [5]. This is, broadly, along similar lines to our work (although we begin with less knowledge of functionality and other requirements). Additionally, the "dark-side scenario analysis", closest to our security testing process, places particular emphasis on risk assessment, whereas the purposes of our own methodology would be to identify specific insecurities relating to an attack goal without looking at the motivations behind it. J3061 [18] also outlines the use of attack trees (in reference to EVITA). The standard also notes that it may only be possible to consider high-level concepts early in the product development cycle. Security analysts or designers could use our methodology as a way of gathering low level requirements for the next design iteration.

In summary, many of the comparative methodologies reviewed above require in-depth knowledge of the system. Our proposed methodology addresses specifically the problem of a black box with many layers of obscurity, all of which may be individually secure, but may exhibit system-wide insecurities.

3 Proposed Methodology

The methodology adopted in this paper is as follows:

Step 1 - Security testing: Since full specifications are generally unavailable, we begin with security testing (more specifically penetration testing) to probe the black box. This is systematised using attack tree methodology. Initial attack trees are first defined relative to an attack goal. These goals can be as low level (flood an open port with data) or high level (denial of service) as needed and tailored to the target interface.

Step 2 - Inferring requirements: Requirements can be extracted from whichever attack proved successful through a process of inference, and is essentially a negation of observed undesirable behaviours found from testing. The determination of security requirements at this stage can be cross-referenced back to the attack tree. This allows for specific insecurities to be addressed as well as separation of security requirements from other types of requirements (known to be useful for interaction analysis [10]).

Step 3 - Suggesting specifications: Once the requirements gathering phase is considered complete, possible specifications could be suggested using a process such as design space exploration. There may be a number of different design choices (and therefore specifications) that could be made to mitigate the threat. These derived specifications could be cross-referenced with other subsets of specifications (such as safety), and where there are contradictions, could help clarify design choices. Where there are no conflicts, the derived security specifications from our process could be added to the overall set of specifications.

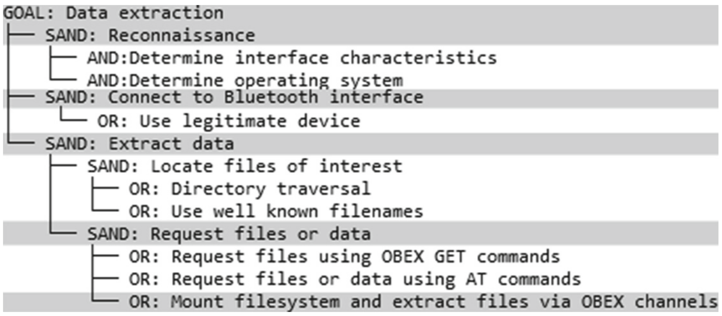
Step 4 - Incorporation of specifications into existing processes: Agreed specifications can be sent down the supply chain. Alternatively, the end user could follow up with in-house model-based design and testing processes. We discuss the latter within the context of our case study (see Sect. 4). The reason for keeping such flexibility is to enable incorporation of this methodology into the wider processes that might be carried out by the end user.

4 Case Study: Automotive Infotainment Unit

For this paper we concentrate on the infotainment unit, where diverse technologies are integrated to deliver functionality such as hands-free communication. We demonstrate the proposed methodology using a case study below. Although this case study came from a single vehicle, it can be reasonably assumed that vehicles of the same make, model and age would share the same weaknesses as production lines are standardised.

Step 1 - security testing: The security testing process was focused on the Bluetooth interface because it is a viable attack vector [15], and because of its ubiquity in cars (an estimated nine million vehicles have implemented this technology [8]). As a vector, it can be used to mount many attacks [4] ranging from denial of service to man-in-the-middle attacks. Implementations can also differ greatly, with various "profiles" available to customise the technology.

The building of the initial attack tree was manually guided, using known vulnerabilities in other Bluetooth applications and surveyed from literature and the National Vulnerability Database [14]. We then evaluated the Bluetooth interface of an automotive infotainment unit using this attack tree (Fig. 1). A number of



```
GOAL: Data extraction
├── SAND: Reconnaissance
│   ├── AND:Determine interface characteristics
│   └── AND:Determine operating system
├── SAND: Connect to Bluetooth interface
│    └── OR: Use legitimate device
└── SAND: Extract data
    ├── SAND: Locate files of interest
    │   ├── OR: Directory traversal
    │   └── OR: Use well known filenames
    └── SAND: Request files or data
        ├── OR: Request files using OBEX GET commands
        ├── OR: Request files or data using AT commands
        └── OR: Mount filesystem and extract files via OBEX channels
```

Fig. 1. Attack tree focusing on extracting data via mounting the filesystem

undesirable behaviours were found, including the ability to mount the filesystem of the infotainment unit and read its contents. This was possible because of the presence of the Object Exchange File Transfer Profile (OBEXFTP) service [1]. We highlight this as an example for the remainder of the paper.

Step 2 - inferring requirements: After having connected to the interface using a legitimate pairing and device (the connection, vehicle or device had not been tampered with in any way), we mounted the file system. Being able to mount the filesystem through Bluetooth could lead to injection of malware, directory traversal and data extraction, manipulation or destruction. As such it is undesirable behaviour, so our inferred requirement from this would be "no unauthorised external agency should be able to see or influence the vehicular operating system's filesystem".

Step 3 - suggesting specifications: Based on the case study attack tree, we could fulfil the requirement above by creating specifications that either (a) remove the ability to request files or data (could conflict with functional requirements); (b) remove the ability to mount the filesystem (may have functional or cost implications) or (c) allow the above, but remove support for extracting, deleting or creating (injecting) files (which would conflict with the required functionality of the FTP server role as specified by Bluetooth SIG [1]).

Step 4 - model-based design and formal verification: Formal analyses of the Bluetooth protocol and its authentication and secrecy properties exist [2]. However, we are not attacking the protocol, but rather probing the larger system in which it resides. This is an example of two components in themselves being secure, but exhibiting insecure behaviour when combined into a larger system. Additionally, all users of the Bluetooth system in this test vehicle are able to use all services offered regardless of who they are. Authentication thus becomes irrelevant. Therefore a more appropriate analysis would be reachability, to demonstrate that such an insecure system state could not be reached through the pathways dictated by the attack tree.

We use the process algebra CSP (Communicating Sequential Processes) to describe a specification of the inferred requirements. We choose CSP because it

channel $pair, connect, advertise\_service, service\_discovery$
channel $service1, service2, displayfs$
channel $obexftp : OBEXFTP\_CMD$
datatype $OBEXFTP\_CMD =$
  $Selectserver|NavigateFolder|MountFS|Push|Pull|$
  $CreateFolder|Copy|Move|Rename|Delete|SetPermission$
$BT = pair \to connect \to advertise\_service \to OFFER\_SERVICE$
$OFFER\_SERVICE =$
    $service1 \to OFFER\_SERVICE$
  $\square\ service2 \to OFFER\_SERVICE$
  $\square\ ..$
  $\square\ obexftp?cmd$
    if $cmd == MountFS$ then
      $displayfs \to OFFER\_SERVICE$
    else $OFFER\_SERVICE$
$USER = pair \to connect \to service\_discovery \to USE\_SERVICE$
$USE\_SERVICE =$
    $service1 \to USE\_SERVICE$
  $\sqcap\ service2 \to USE\_SERVICE$
  $\sqcap\ ..$
  $\sqcap\ obexftp!MountFS \to displayfs \to USE\_SERVICE$
$IMPL = BT \underset{\alpha_{BT} \cap \alpha_{USER}}{\parallel} USER$

Fig. 2. Small illustrative model of the OBEXFTP service

Suggested (property-oriented) specification:
$SPEC = CHAOS_{\alpha_{IMPL} \setminus \{displayfs\}}$
Refinement assertion:
assert $SPEC \sqsubseteq_{traces} IMPL$
Result: FAIL

Fig. 3. Example specification and verification

is able to represent and combine the message passing choreography expected by individual components. A complete introduction may be found in [17].

The specification for the Bluetooth FTP is available [1] and so we developed a small illustrative CSP model (Fig. 2). We then developed a suggested specification from our inferred requirement, such as never displaying the filesystem (see Fig. 3) which fails during the verification process with trace $\langle pair, connect, service\_discovery, advertise\_service, obexftp.MountFS, displayfs \rangle$. If, however, we removed OBEXFTP, and assuming none of the other services offered the ability to mount a filesystem, it would verify correctly.

We use this exercise to show that the inferred requirement is not met by the standard FTP specification (that a server must respond to a request from a client for "Folder Listing Objects") and is, in fact, contradictory. Thus, any attempt to remove support whilst still maintaining the profile would be breaking Bluetooth's specification.

Here, the model and example specification is simplistic enough to make it self-evident that the removal of OBEXFTP would allow for successful verification. This exercise would add value provided: (a) the systems are sufficiently complex; (b) we can create a more accurate model of the system under investigation, or (c) there is more than one path to mount the filesystem (or, more generally, to achieve any other undesirable behaviour).

5 Discussion and Conclusion

Our methodology is suited for tiered supply chains, as there is no need to have complete specifications of the integrated item for security testing. It also reflects real world security issues that have arisen through the testing process. The attack tree methodology allows for systematisation and traceability, especially where design choices are concerned. These choices could also be cross-referenced against scenarios that were posited in attack trees but were not tested. Any security requirements gathered can be kept separate for interaction analysis and allows for reasoning about alternatives. The formal exercise could allow for clarification of ambiguities, and using a verifier leads to a higher level of confidence in the resulting design (albeit dependent on the model constructed). Limitations include the fact that the initial creation of the attack tree is manually guided although domain expert input in reviewing the tree and repeated testing over more vehicles would mitigate this. There is also a one-off cost of building these trees, although reuse is possible in future testing processes. As testing scope expands, trees could also become crowded, and so tree navigation will be essential. Problems with scalability could also be mitigated using mechanical tools such as design space exploration. Furthermore, the data available to construct the model at the end of the process directly impacts the quality of the model created.

In this paper, we have presented a methodology for securely combining third party components into a wider system and applied it in the context of an automotive head unit using the Bluetooth interface. We have found weaknesses through structured security testing, and using the case study of being able to mount the filesystem through Bluetooth, we demonstrated how to infer security requirements and suggest specifications. We have also recommended follow-on processes that we envisage end users would find constructive in strengthening the security of their systems. Future work would include refining the process by applying the methodology to a more significant case study, with different attack goals. Through both of these, we also aim to acquire enough information as to be more concrete with regards to formal processes. Ultimately, we wish to position this methodology in a larger design process such as that espoused by standards such as J3061.

Acknowledgements. The authors would like to thank Olivier Haas (Coventry University) and Alastair Ruddle (HORIBA MIRA) for valuable comments.

References

1. Bluetooth SIG Inc.: Bluetooth Specification: File Transfer Profile (FTP) (2012)
2. Chang, R., Shmatikov, V.: Formal analysis of authentication in bluetooth device pairing. In: Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis, p. 45. Wroclaw, Poland (2007)
3. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of 20th USENIX Security Symp. pp. 77–92. USENIX Assoc., San Francisco, August 2011
4. Dunning, J.P.: Taming the blue beast: a survey of bluetooth based threats. IEEE Secur. Priv. **8**(2), 20–27 (2010)
5. EVITA Project: Deliverable D2.3 - Security requirements for automotive on-board networks based on dark-side scenarios. Technical report (2009)
6. Fuchs, A., Rieke, R.: Identification of security requirements in systems of systems by functional security analysis. In: Lemos, R., Gacek, C., Casimiro, A. (eds.) Architecting Dependable Systems VII. LNCS, vol. 6420, pp. 74–96. Springer, Heidelberg (2010)
7. Gordon, D., Stehney, T., Wattas, N., Yu, E.: System Quality Requirements Engineering (SQUARE) Methodology: Case Study on Asset Management System. Techniacl report Carnegie Mellon University, Pittsburgh, May 2005
8. GSMA: Connected Car Forecast: Global Connected Car Market to Grow Threefold within Five Years. Technical report, GSMA (2013). http://www.gsma.com/connectedliving/wp-content/uploads/2013/06/cl_ma_forecast_06_13.pdf
9. Idrees, M.S., Roudier, Y., Apvrille, L.: A framework towards the efficient identification and modeling of security requirements. In: Proceedings of the 5th Conference on Network Architecture and Information Systems, pp. 1–15. Menton, France, May 2010
10. van Lamsweerde, A.: Elaborating security requirements by construction of intentional anti-models. In: Proceedings of 26th International Conference on Software Engineering, p. 10. IEEE Computer Society, Edinburgh, May 2004
11. Liu, B., Shi, L., Cai, Z., Li, M.: Software vulnerability discovery techniques: a survey. In: Proceedings of the 4th International Conference on Multimedia Information Networking and Security. IEEE, Nanjing, China (2012)
12. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
13. Midian, P.: Perspectives on penetration testing - Black box vs. white box. Netw. Secur. **2002**(11), 10–12 (2002)
14. National Institute of Standards and Technology: National Vulnerability Database
15. Oka, D.K., Furue, T., Langenhop, L., Nishimura, T.: Survey of vehicle iot bluetooth devices. In: Proceedings of the IEEE 7th International Conference on Service-Oriented Computing and Applications, pp. 260–264. IEEE, Matsue, Japan, November 2014
16. Opdahl, A.L., Sindre, G.: Experimental comparison of attack trees and misuse cases for security threat identification. Inf. Softw. Technol. **51**(5), 916–932 (2009)
17. Roscoe, A.: Understanding Concurrent Systems, 1st edn. Springer, London (2010)
18. SAE International: J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (2016). http://standards.sae.org/j3061_201601/
19. Schneier, B.: Attack trees: modeling security threats (1999). http://www.schneier.com/paper-attacktrees-ddj-ft.html