Product Lines Can Jeopardize Their Trade Secrets
Mathieu Acher, Guillaume Bécan, Benoit Combemale, Benoit Baudry, Jean-Marc Jézéquel

To cite this version:
Mathieu Acher, Guillaume Bécan, Benoit Combemale, Benoit Baudry, Jean-Marc Jézéquel. Product Lines Can Jeopardize Their Trade Secrets. 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Aug 2015, Bergame, Italy. 10.1145/2786805.2803210. hal-01234342

HAL Id: hal-01234342
https://inria.hal.science/hal-01234342
Submitted on 1 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Product Lines Can Jeopardize Their Trade Secrets

Mathieu Acher, Guillaume Bécan, Benoit Combemale, Benoit Baudry, and Jean-Marc Jézéquel
Inria - IRISA, University of Rennes 1, France

ABSTRACT

What do you give for free to your competitor when you exhibit a product line? This paper addresses this question through several cases in which the discovery of trade secrets of a product line is possible and can lead to severe consequences. That is, we show that an outsider can understand the variability realization and gain either confidential business information or even some economical direct advantage. For instance, an attacker can identify hidden constraints and bypass the product line to get access to features or copyrighted data. This paper warns against possible naive modeling, implementation, and testing of variability leading to the existence of product lines that jeopardize their trade secrets. Our vision is that defensive methods and techniques should be developed to protect specifically variability – or at least further complicate the task of reverse engineering it.

Categories and Subject Descriptors

D.2.0 [Software Engineering]: Protection mechanisms;
D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement—Reverse engineering and reengineering

Keywords

Product lines, reverse engineering, ofuscation, security

1. INTRODUCTION

The engineering of a product line makes it possible to offer numerous configuration options (or features) and deliver unique products (or variants) to customers. Substantial profit is expected in terms of customer satisfaction, mass customization, market presence, etc. But what do you give for free to your competitor when you exhibit a product line? Hopefully nothing: the technical, software realization and the underlying business information of a product line should generally remain hidden to avoid losing some economical edge. In reality, it may happen that product lines jeopardize their trade secrets. For instance, an attacker or a competitor can identify hidden constraints and bypass the product line to get access to features or copyrighted data.

As a first concrete example, let us consider an online newspaper. (We use a real-world example, see Section 2.1 for further details.) This newspaper freely delivers online news and articles to readers. In addition to this free content, there is a protected access for paying subscribers that allow them to read brand new content a few hours before it is made public. However, a naive implementation in the website of the newspaper allows a regular reader to access protected articles without paying. The code impacted is given after:
their trade secrets. There are two kinds of trade secrets. First, the way a product line is realized: if discovered, the technical advantages are lost, a third-party can further re-exploit domain artefacts, etc. Second, the confidential information of a product line: it may be copyrighted content, a marketing practice, etc. The two secrets are related: the understanding or reverse engineering of the technical realization is usually necessary to collect sensible information.

We present case studies in which the discovery and understanding of variability is possible and can lead to severe (economical) consequences. Section 2.1 further details the case of online newspapers, showing other variability-based bugs. Section 2.2 describes an online generator of video variants in which protection of copyrighted data matters. Section 2.3 points out the protection issues faced by Web configurators.

As a result, it is not enough to model, implement, and test a product line. Our vision (hence the second major point of the paper) is that defensive methods and techniques should be developed to specifically protect variability and configurations. We call to further investigate the protection perspective onto software variability, a topic quite absent in the software engineering and product line literature. Numerous techniques for software protection (e.g., code obfuscation [4]) have been considered, but the specificities of variability and configurations raise novel challenges. A malicious attacker should have difficulties to build mental abstractions, to identify and reason about variability, and to navigate into the configuration space. Otherwise the reverse engineering and the re-engineering of the product line is highly facilitated and can even be automated. Section 3 summarizes our findings, discusses potential techniques for protecting variability, and outlines a research roadmap.

2. PRODUCT LINES CAN JEOPARDIZE TRADE SECRETS

2.1 Online Newspapers (Cont’)

The first example in the introduction is based on a real wrong design decision\(^2\) that has been fixed afterwards. We now describe another problem in the same domain. For confidential reasons, we call it fakenewspaper hereafter. The website is separated in two domains (1) fakenewspaper.com gives a limited access to public articles: (2) subscriber.fakenewspaper.com gives a complete access to paying customers. In complement to providing early access to new articles, the variant of fakenewspaper for subscribers provides additional services (e.g., easy-reading option, limited amount of advertising). As in the previous example, the verification is done on the client side. When a visitor accesses subscribers.fakenewspaper.com, a JavaScript checks whether she is a member. In case the user is not a member, the page is redirected to fakenewspaper.com.

Why protecting variability? By deactivating JavaScript, a regular (non paying) reader can access articles that should be restricted to subscriber.fakenewspaper.com. An outsider can even implement a script that automates the task of finding the complete text in subscriber.

...
Figure 1: Video generator: modularity and variants

2.3 Web Configurators

A Web configurator provides an interactive graphical user interface that guides the users through the configuration process, verifies constraints between options, propagates user decisions, and handles conflictual decisions [1]. Configurators are used in installation wizards, preference managers, and extensively used in product lines. The database maintained by Cylde is a striking evidence with 900+ Web configurators coming from 30+ different industry sectors [6].

A very simple excerpt of a real-world Web configurator is depicted in the Figure below. We can observe that the selection of Diesel has lead to the automated selection of EDC6. (In fact, some other equipment options have been previously selected; one can select Diesel with EDC7 or BVM in some other configuration settings.) That is, a customer has not chosen EDC6; the configurator has imposed some constraints of different natures: technical/engineering constraints, aesthetic constraints, marketing constraints, etc.

Why protecting variability? Products, options, and the underlying constraints a configurator is in charge of are key information of an organization. Such information is particularly interesting from the perspective of (online) market intelligence (MI) (also called competitive intelligence). MI can be defined as the “information relevant to a company’s markets, gathered and analyzed specifically for the purpose of accurate and confident decision-making in determining market opportunity, market penetration strategy, and market development metrics.” Lixto, a company offering data extraction tools and services for MI, showed that it is technically feasible to acquire and exploit unstructured and semi-structured data in several case studies (e.g., in the domain of computers and electronics consumer goods [2]).

Most information on product pricing, availability, options and constraints is potentially available on Web sales configurators. Specifically, competitors can use this information (1) for getting a comprehensive overview of the options and constraints in the market; (2) to be (continually) informed about strengths and weaknesses of other competitors’ product lines; (3) to publicly reveal a certain superiority or marketing practice, etc.

Web data extraction systems [2, 5] can be specialized for acquiring configurators’ information. Early attempts shows that reverse engineering Web configurators is feasible [1]. Static analysis techniques can locate templates of options and some constraints in a Web page. Combined with crawling techniques for deep navigation and dynamic content pages, there is the potential to fully gather relevant information.

In case the static and dynamic analysis of variability can be seamlessly realized, there is a risk for companies developing Web configurators to reveal trade secrets.

3. PROTECT VARIABILITY!

We observe that the case studies are sharing similar classes of security and protection issues. We identify three kinds of vulnerabilities (related to positive variability, negative variability, or configuration space) and leading to the possible leaks of trade secrets. We draw a research roadmap highlighting four directions (RD1, RD2, RD3, RD4).
Protection of positive variability (RD1). Voelter et al. describe variability implemented with compositional approaches as positive variability since variable elements are added together [7]. It is the case of the video generator case study (see Section 2.2) in which video sequences are assembled to build a video variant. It may also be the case in Web configurators (see Section 2.3) in which options are added on-the-fly depending on some user choices.

We have shown in Section 2.2 that an attacker can too easily reverse engineer the product line with a clean modularization design. The vulnerabilities are coming from the identification of the modules and their direct mapping in terms of features in the variability model. From this identification, an attacker can infer how these modules can be composed together to re-engineer the product line (i.e., by positive variability). It should be noted that the positive variability (and the underlying issues) apply either at the data level (e.g., videos) or at the implementation level (source code) of the product line. Two main approaches are then possible to prevent the discovery of trade secrets.

First, source code deconstruction, such as control flow degeneration and data flow disturbance, are essential obfuscation techniques [4]. An open challenge is to develop techniques for obfuscating specifically the variability and modularity in the source code or data. A second approach is to obfuscate the mapping between features/options and the corresponding artefacts. For instance, a one-to-one mapping may be too easy to identify and understand. The challenge is to develop innovative techniques, ideally non intrusive for product line developers and agnostic to a domain, for diversifying the mapping. The two approaches can be used independently or in combination, depending on the product line characteristics.

Protection of negative variability (RD2). Some product lines exhibit all their functionality and content once and for all. They use negative variability (as opposed to the previous positive variability) in which all different variants are expressed; the variants are activated depending on some conditions [7]. It is the case of online newspapers (see Section 2.1). It may also be the case in Web configurators (see Section 2.3) in which options are hidden or depicted on-the-fly depending on some user choices. The source code of such configurators already contains the content for activating/deactivating options typically through JavaScript. Negative variability cannot be accused of being the root cause of vulnerabilities. However, it is necessary to either: (1) improve the mechanism used to remove or activate some variants. For instance, access controls (e.g., as at the server side) or obfuscations can be considered; (2) obfuscate the pre-defined variants. For example, in the case of online newspapers, the content of members’ articles can be encrypted.

An interesting research direction is to determine whether (and if yes, when) negative variability presents more security guarantees than positive variability (or the other way around). Product lines can indeed use the two kinds of variability (e.g., as in Web configurators).

Barriers to master the configuration space (RD3). Understanding a configuration set may have an interest per se since trade secrets are hidden there. For example, the video generator of Section 2.2 has a strategy for generating some frequencies of features. The idea is that some features corresponding to some video sequences are rarely activated (e.g., in 0.1% of configuration) for surprising the visitor. As another example, Web configurators exhibit options with marketing or technical constraints. In the two examples, the ability of an attacker to crawl the configuration space is the key for discovering trade secrets. A comprehensive visit is the worst situation since the extracted knowledge is then complete. Another threat of a (comprehensive) mastering of the configuration space is that attacker can experiment the effects the configurations have on the product line. It is one of the basis to understand or guess the underlying implementation of a product line. In the video generator, setting a configuration leads to a new video variant that can be then analyzed. A comprehensive visit is again the worst situation since all corresponding variants and related artefacts are then accessible.

The challenge of RD3 is to develop barriers to limit the exploration of the configuration space. For instance, mechanisms for blocking IP addresses can be considered in case many requests for crawling the configuration space are observed (see Section 2.2). Many specific factors can influence the definition of a politics of configuration access.

RD4: Cost-benefit tradeoffs. On the one hand, protecting variability and configurations has admittedly a cost. The technical or management effort can be more or less important – from a drastic change in the design of the product line to small increments to re-enforce access controls. On the other hand, the trade secrets an organization has to protect and the possible consequences highly vary. A trade secret can give access to very few non-members, but can also lead to lose any competing advantage.

Hence a tradeoff has to be found. The importance of trade secrets a product line can jeopardize should justify the investments required to develop and deploy protecting mechanisms. A spectrum of more or less sophisticated techniques can thus emerge. An ideal solution (hence a challenge) is to let product line developers follow their usual methods while guaranteeing adequate security.

Concluding remarks. In this paper, we provided evidence that product lines can jeopardize their trade secrets. Our vision is that it is urgent to cross-fertilize the research results in software product lines and security. Any software systems may come up against security/protection issues, yet the specificities of handling variability/configurations raise novel and difficult challenges. We mainly took the perspective of a defender; the research can be considered from an attacker point of view as well (though ethics and legal aspects have to be defined). For concluding, we formulate the idea that software variability of product lines, as a key competing advantage and first-class citizen, should itself vary to complicate the task of an external attacker.

4. REFERENCES