Designing Adaptive Deception Strategies

Cristiano De Faveri, Ana Moreira

NOVA LINCS, Department of Computer Science
Faculty of Science and Technology
Universidade NOVA de Lisboa
c.faveri@campus.fct.unl.pt, ammfct.unl.pt

Abstract—Deception-based defense is the process by which actions are intentionally employed to cause misrepresentation and induce erroneous inferences on attackers. Deception can be employed in different levels of computation, from network to application-level, which demands careful planning and coordination between multiple strategies and tactics. Despite of advances on using deception in computer defenses, ad-hoc approaches are still used for their design. As a result, deception is realized essentially as single tools or as entire solutions repackaged as honeypot machines. In this paper, we propose a model to specify coordinated deception tactics based on adaptive architectures. Our contributions rely on a deception-based defense life-cycle approach integrated in a software design process, including a model to specify coordinated deception strategies. The feasibility of the proposed approach is shown via an example where a deception strategy is designed for a smartphone application that synchronizes data with a central database.

Index Terms—computer security, deception, software design

I. INTRODUCTION

The increasing evolution of software-intensive systems in number of applications and services, challenges engineers to build secure solutions. These systems usually operate in a complex eco-system under partially-trusted settings and different third-party components, which accentuate the systems’ exposure to adversaries. Additionally, advanced persistent threats (APT) are continuously increasing [1], [2]. APT represents a wide range of sophisticated reconnaissance and information-gathering tools, as well as attack tools and methods, used by skilled and well-resourced criminals to gain entry to networks and front-end servers. APT is hard to detect and defend against, as evidenced by the ever-rising number of successful attacks and the increase in their severity [3]–[5]. Estimates show that cybercrime annual cost to the global economy is above $400 billion [6].

Security is primarily devoted to protect valuable assets, requiring well-balanced security mechanisms to maintain costs affordable while decreasing the success of attacks. Despite of the increasing advances in security mechanisms to protect systems’ assets, the complexity of modern software systems aligned with the increasing time-to-market pressure, simply preclude perfection. Even with all controls enabled by organizations, attackers have proven to go undetected or caught while successfully achieving their goals [7], [8]. To complicate matters, assets may evolve dynamically based on environment changes, demanding security controls to adapt to situations not always foreseen at design-time. Therefore, additional security barriers are necessary to increase the detection success and response with the capability to carefully monitor and study adversaries.

Studies on evolutionary biology show that deception plays a key role in creatures’ evolution, representing an advantage over their competitors [9]. This idea has been mimicked in the virtual world where deception has been typically employed as a second line of defense to prevent, detect, and respond to offensives. Deception- and decoy-based techniques aim at altering the perception of attackers by using negative information and/or deliberate actions in favor of defense. In cyber defense, deception has been investigated for more than two decades to early-warning and advanced security surveillance tool, minimizing the risks from attacks on IT systems and networks [10]–[12]. However, approaches to incorporate deception in computer defenses are still not systematically coordinated with the software design process. The result is that deception appears in isolated tools (e.g. honeyfiles [13] and honeywords [14]) or else as an entire repackaged solution (e.g., a honeypot machine [15]).

As pointed in [16], incorporating deception in the design of software gives system defenders an increased ability of obtaining information from an attempt of compromising a system. In this sense, deception planning requires careful application of multiple techniques to produce plausible deceptive stories to the opponent [17]. In particular, as the number of tactics and methods increase, strategies to coordinate multiple methods, validate multiple deception channels, and monitor the effectiveness of integrated operations become a challenge [16]. In this paper we:

• propose a process model to specify coordinated deception tactics as part of an adaptive life cycle that monitors and modifies deception strategies according to environment changes and attackers interactions;
• describe activities to design deception tactics involving the definition of goals, monitoring channels, bias exploitation, deception stories, and risks;
• define functions performed during runtime, including adaptation, to monitor attacker interactions and the risks associated with deception strategies, and decide for distinct strategies to deceive attackers;
• use an example to show the feasibility of our proposal to create coordinated deception tactics.
This paper is organized as follows: Section II offers an overview of deception theory and its application on cyber defense. Section III presents a vision of deception defense life cycle as part of an adaptive system, and Section IV offers the main modules involving an infrastructure to support deception analysis and system reconfiguration. Section V describes a model to design deception tactics and strategies. Section VI presents some examples to illustrate the application of our approach. Related works are discussed in Section VII, and Section VIII draws conclusions and points directions for our future work.

II. DECEPTION-BASED DEFENSE

Deception represents the process by which actions are deliberately employed to cause misrepresentation and induce erroneous inferences [17]. It has an extensive history in the military arena, where successful operations largely depend on the correct assessment of the adversary military forces. War strategies expose the idea of asymmetry, in which advantage situations are reached by effective actions on the battlefield. Such advantages are frequently gained by using deceptive tactics to create asymmetry and manipulate the enemies to reduce their effectiveness [18]. This causes misallocation of resources, so that attacks can be either planned for favorable time and place or even avoid [19]. The enemy is pushed into a pre-positioned ambush that distracts them while a much larger force moves to attack from the rear [20]. Much like the conventional warfare, strategic deception in cyberspace has been applied to offensive and defensive purposes. Many deceptive tactics employed in traditional warfare can be rephrased and used in the cyberspace.

A. Deception Elements

The broad concept of deception encompasses several subsidiary ideas and coordinated actions synchronized with the objectives of deceivers. Deception goes beyond the act of “lying” to take some advantage over a target. It also stresses the practice of keeping secret the existence of some truth by negating access or withholding the information. [17] discusses two strategic deception tactics — simulation, the act of showing the false, and dissimulation, hiding the real—, and proposes a taxonomy of deceptive techniques using these tactics, compounded by mimicking (imitating something/someone's identity to appear legitimate), inventing (making up of information), decoying (distracting the attention of targets from the transaction), masking (concealing relevant information about the item), repackaging (making a relevant object appear to be something it is not), and dazzling (hiding information regarding an item).

Dunnigan and Nofi present a taxonomy containing nine categories of deception techniques: concealment (hiding your forces), camouflage (hiding your troops by artificial means), false and planted information (fake information that harms the enemy and helps you), lies (when communicating with the enemy), displays (techniques to make the enemy see what isn’t there), reuses (tricks, such as displays that use enemy's equipment and procedures), demonstrations (making a move that implies imminent action, but not followed through), feints (like a demonstration, but you actually make an attack), insight (deceive the opponent by out thinking him) [21].

Rowe presents another taxonomy, based on linguistics case theory and artificial intelligence, which groups deception into the following dimensions: space, time, participant, causality, quality, essence, and speech-act theory [22]. This taxonomy does not consider deception techniques. Instead, deceptive actions have associated concepts that help particularizing them, and these are conveyed in language by modifiers, prepositional phrases, participial phrases, relative clauses, infinitives, and other constructs. For example, space is categorized into direction (of the action), location-at (where something occurred), location-from (where something started), location-to (where something finished), location-through (where some action passed through), and orientation (in some space).

B. Cognitive aspects of deception

To be successful, deception must be targeted to influence the thinking of the adversary. Understanding how the thinking process occurs within the human abilities of judgment is critical to increase the chances of well-succeeded deception operations. Understanding how humans perceive and process information to achieve analytical judgments remains a key challenge. Psychologists have delved into how the perception of the world and of other subjects diverge from reality in patterns that could be detectable and explained. These patterns are generally referenced as “biases”, which can influence decision-making, beliefs, and behavior [23]. Thompson et al. [24] discuss three major groups of biases:

- **Personal and Cultural**: Personal biases are constraints that arise from individual’s past experiences. Cultural biases refer to thinking constraints acquired during evolution from widely held beliefs, practices, or cognitive styles that characterize one’s specific social environment.
- **Organizational**: Organization biases are restrictions imposed to individuals by the local information, traditions, and goals of the organization they are part of. In this level, deception tactics can be designed taking into consideration insider attackers (current and former employees, for example).
- **Cognitive**: Cognitive biases are intrinsic to our nature, referring to patterns of deviation in judgment, whereby inferences may be drawn in an illogical fashion [26], [27]. Cognitive biases may lead to perceptual distortion, inaccurate judgment, illogical interpretation, or what is broadly called as irrationality [28].

Deceptive tactics exploiting personal, cultural or organizational biases tend to be more effective as they exploit a particular cognitive vulnerability of an individual or group of
Deception techniques have been investigated for a long time as information system defense. Earlier documented works showing the use of deception include "The cuckoo’s egg" from Clifford Stoll [10] and "An Evening with Berferd" [11], in which the authors discussed how they interacted with an attacker by providing fabricated responses. These reports culminated in the first ideas of honeypots. Honeypots are resources primarily designed to entice attackers and learn their tactics [15]. Typically, honeypots are categorized into research and production. A research honeypot aims at providing data on the methods used to attack systems and present little risk of propagating an attack. On the other hand, production honeypot targets the protection of the organization while distracting the attacker. However, it presents more risk of propagating an attack. The deception toolkit, released in 1997, is one of the early tools designed to apply deceptive techniques for cyber defense (DTK) [12]. This tool allows the creation of simulated Unix-based services configured by a deception-specific finite state-machine language to determine the level of emulation of legitimate services. Honeypots are often grouped in clusters called honeynets [30]. Honeynets are research honeypots but without emulating services, i.e., the network is built using real systems and applications. At client-side, honeypots have been proposed to identify malicious websites that target client application vulnerabilities [31]. Similarly, mobile honeypots focus on deceptive techniques for threats on mobile devices [32]–[34].

Deceptive techniques have also been investigated into finer-grained digital entities. The traditional manifestation of honeypots supposes a computer, or some physical resource for the attacker to interact with. Whereas this vision targets deceiving network attacks, it represents a restricted notion, by definition, of the honeypots potentialities. Honeypots that are not computers are called honeytokens [35], a term emerged in the beginning of 2003 to characterize the use of honeypot techniques in any digital entity. A honeytoken can be any sensitive data in a database, such as credit card numbers, passwords and salaries, files like spreadsheets, presentations, or even a bogus login service.

Typical honeytoken techniques include the use of desinformation, excuses and delay responses. Desinformation is the act of planting false information into the system to cause confusion or lead attackers astray. Desinformation is also a mean to detect illegitimate actions. For example, honeywords is a method to improve the security of hashed passwords by using false passwords (the honeywords) on a file associated with each user’s account [14]. An approach with similar objectives is presented by Almeshekah, M. et.al. [36]. Honeyfiles are decoy files deliberated installed on a server, containing appealing names for attackers, like allpasswords.txt [13]. Honeygen is a method designed to create honeytokens in general relational databases [37]. In [38], the authors leveraged the honeygen idea by considering honeytokens as cross-cutting concerns, and proposes honeytokens generation using aspect-oriented technology [39]. White proposed a method for creating honeytokens representing personal data such as names, addresses, and social security numbers [40]. The Decoy Document Distributor (D³) system was developed as a web-based service that detects file misuse [41]. To extend the exposition of honeypots, the Google Hack Honeypot³ leverages the purpose of search engines by emulating a vulnerable web application to be indexed by search engines.

While most of these approaches are designed to be application-agnostic, information systems can also deceive to protect themselves against dangerous actions. Useful lies can give excuses for resource denial, like saying "the communication with our central server is broken" in response for a command request. While general excuses exploit common resources of systems (such as communication channels, network issues, authentication methods, and database interaction), domain-specific excuses can coop with more realistic excuses. A financial system delivering an excuse “Your account has been locked. Please contact the bank.” may cause in the attacker’s mind the perception that the attack was successfully accomplished. Well-designed excuses have the advantage of keeping attackers distrustful, worried of being detected. Thus they may keep wasting time trying to use the system resources.

Delay responses is a deceptive tactic used when the defender requires extra time to assemble a defense. Delaying can be achieved by slowing down a system response, but also giving to attacker more extra questions to answer or additional information to read before they can proceed. Delay responses can also be integrated to excuses to advise attackers to wait for some event before moving ahead.

III. A LIFE CYCLE VIEW FOR DECEPTION STRATEGIES

Specifying coordinated deception tactics requires the identification of the objectives of the deception strategy and an understanding of how the tactics relate to each other. This allows the analysis of adverse situations that may arise during the implementation of a deception strategy. In this section, we introduce an approach to build and execute coordinated deception tactics as a realization of an adaptive system that evolve strategies in response to attackers’ behavior and environment changes. Figure 1 lists sets of activities to handle deception strategies during the design, runtime and adaptation phases.

At design-time, deception tactics are specified and implemented following different techniques adopted by the software

³Despite of the name “Google”, the technique is available for any search engine.
engineer and the security engineer. Deception defense tactics may involve risks that need to be identified during design-time and constantly evaluated during runtime, so that the system can adapt to mitigate them. The adaption phase is responsible for analyzing deception channels and adapting (through reconfiguration) the strategies and tactics according to a set of criteria defined at design-time.

A. Design-time

During design time, five main activities are performed: design deception strategies (D1), identify risk (D2), define monitoring deception channels (D3), define deception metrics (D4), and define adaptation policies (D5). Designing deception strategies includes specifying a repertory of deception strategies containing deception tactics and deception stories. To understand how a deception tactic has been exploited, deception monitoring channels are defined. A monitoring channel provides information about action/inactions performed by adversaries in a deceptive component. Techniques used for monitoring vary on intrusiveness, from actively instrumenting source or intermediate code to passively reading system logs. The definition of metrics allows to assess the effectiveness of a deception tactic. These metrics are analyzed by the adaptive mechanism as part of the decision process for reconfiguration. The implementation of deception tactics may introduce risks, which need to be identified, analyzed and accepted by the defender before being activated. Finally, defining adaptation policies identifies the rules used during adaptation. Typically, eliciting adaptation requirements follows the 5W1H structure, composed of six questions [42]: what, where, who, when, why and how. What refers to what attributes or artifacts of the system can be changed through adaptation actions, and what needs to be changed in each situation. Where identifies artifacts and level of granularity that is causing the need for adaptation. Who addresses the level of automation and human involvement in self-adaptive software. When identifies temporal issues of adaptation, such as when adaptation is required, how often the system is required to adapt, and which strategy (predictive or reactive) will be used to perform adaptation. Why addresses the reasons of building a self-adaptive software application. Finally, how refers to how adaptation takes place, including which actions are necessary, the order of the actions, the costs and aftereffects, and a rollback plan.

B. Runtime

During runtime, deception monitoring points previously defined during design-time are observed (R1), and relevant information is collected to an incident database. An incident corresponds to an event that is part of expected interaction with a deceptive tactic. For example, an attempt to guess a password (multiple wrong attempts), an access to a fake service, and an abnormal parameter input are typical events to be recorded to an incident database. Report deception exploitation (R2) refers to the analysis and reporting of incidents to administrators or to primary level of protection mechanisms. Risks require continuous monitoring (R3) to guarantee security goals are preserved and deception tactics are exploited as expected. Continuous risk assessment outcomes are used in the adaptation phase as part of decision-making process.

C. Adaptation

Adaptation is the process in which deception strategies are analyzed (A1) in terms of effectiveness, and risks rates are considered to determine whether the system should be reconfigured. Incidents recorded during runtime are analyzed in the adaptation phase, that can decide for reconfiguration (A2). Reconfiguration is performed in terms of parameters (parametric reconfiguration) or system structures (add/remove components). Evolution is triggered whenever any optimal solution is found during the decision process. In this case, the system should expose the reason a solution was not found to assist engineers to evolve the tactics. Evolution requires modifications on current deception tactics and/or the design of new tactics to be incorporated to the system, while adaptations can occur automatically for the anticipated situations at design-time.

IV. ADAPTIVE DECEPTION PLATFORM

To support adaptation activities, we propose the use of sensors and effectors to monitor deception components and the environment, while reconfiguring the system in response to the increase of associated risks or a particular action/inaction performed by attackers. Whereas system components are those designed to perform required functions in the system, a deception component is any resource in the system with deceptive purposes. Figure 2 illustrates the main elements of the proposed platform. We can see the platform as an application of MAPE-K framework [43], where the Deception Monitor is responsible for collecting information about the deception components and the environment, and feed the Incident Database. Monitoring is performed through channels defined to collect data about actions/inactions performed by
the deceived. Channels can be generic for any simulated component or specific for a particular component. The former are intrinsically provided by the deception infrastructure. The latter requires to be designed on the deception component and made available to be linked to the monitoring mechanism. Deception Action Analyzer is the module responsible for analyzing the incidents and deciding for reconfigurations based on these incidents. Similarly, Deception Risk Analyzer is responsible for analyzing the associated risks and deciding for changing the deception strategy, including cease it. Deception effectors are responsible for executing the reconfiguration and for recovering the system to the previous state in case of intermediate failures during adaptation.

![Deception Platform](image)

**Fig. 2:** Proposed deception platform

### V. Deception Model

To support the design of deception defense, Figure 3 proposes a model with the main concepts to be employed in multiple activities of the deception strategy life cycle (Section III). Deception tactics represent a conceptual notion realized by a set of deceptive functions and system states, with the intention of causing expected reactions on adversaries. A deception tactic may contain decoy elements, i.e., some characteristics in the tactic that cause adversaries enticement and consequent engagement to react in some direction. A tactic may employ multiple deception techniques within simulation and dissimulation dichotomy. Functions and states of the system are described in terms of deception stories.

![Deception Strategy Model](image)

**Fig. 3:** An excerpt of the deception strategy model

A deception story establishes behavioral aspects of deception strategy and may involve many tactics. Also, a deception story may exploit one or more biases, such as personal or cognitive. Thus a deception strategy is compounded by a set of deception tactics and deception stories used to achieve one or more goals. We associate biases in stories because, depending on how we ‘tell’ the story, different biases can be exploited. A deception strategy is monitored by channels that collect information about adversaries interactions. Metrics are defined to validate the effectiveness of a deception strategy, so parameters can be established to support the adaptation process (see Section III-A). Finally, deception strategies may involve risks that need to be identified and properly managed. Risks are identified in two directions: i) the security risk a given tactic or story can impose in the system and ii) the risk of a deception tactic being acknowledged by an attacker. The former implies identifying, monitoring and mitigating risks that jeopardize assets by violating security goals of the system. Deception tactics must be designed in such way that security properties of the system are preserved; otherwise, the deception tactic would be considered risky for the operation. The latter treats the risks of a deception tactic being discovered by the attacker. Attackers who recognize the use of deception may assemble stronger offensive to the system in retaliation, raising the overall security risk.

### VI. Designing Deception Strategies

Consider an illustrative scenario where an application designed for a logistic company is installed on the employees’ smartphones. This application saves flat files containing sensitive information about operations and log files containing transactions, warning, errors, etc. Also, the application communicates with a remote database to synchronize information frequently. We would like to create a deception strategy to recognize illegal accesses to the employee’s smartphone file system, i.e., whether an attacker has overcome the first line of security (protection) and reached the smartphone’s files. For this scenario, we will establish a deception strategy in which a bogus database will be used to trigger security alerts when accessed. Table I illustrates goals, monitoring channels, metrics and risks associated with this deception strategy. The strategy aims at identifying spurious access to the file system but also diverting attackers with bogus data. We set ‘authentication functionality’ and ‘command sent to the DBMS’ as monitoring channels. Also, the number of authentication attempts, the number of successful accesses, and the number of DML and DDLs commands sent to the DBMS are metrics deemed in this strategy. General risks are identified in terms of deception plausibility and average system security. Attackers identifying data as fake jeopardize the application of the strategy, and there is a risk that sensitive data originated from original database has been loaded into the bogus database. This depends of how the building of fake database is considered during the its implementation. Also, if the bogus database gives somehow a passage to attackers access the original database, the deception strategy is not only deemed worthless but harmful to the system. The process...
of identifying risks is iterative, where refinement may occur during the design of tactics and stories.

Table II illustrates the tactics associated with the deception strategy. The first tactic (T1) involves the creation of a fake database containing bogus accounts associated with vulnerable passwords. We use the deception technique classification proposed in [17], in which T1 is classified as a repackage (since we are Repackaging the original database as a normal database). The second tactic (T2) covers the building of bogus database tables, quite similar to the original tables to keep a plausible story. However, we introduce Inventing in the technique to open the possibility of creating new tables for enticement purposes. Similarly, database items (T3) are fabricated to divert attackers in retrieving bogus data. The last tactic (T4) refers to the instrumentation of the log file to add decoys that point to the fake database. These causes confusion on attackers who does not recognize false and true records in the file. Of course, instrumenting the log file requires a mechanism to map which record is part of the deception tactic. All of described tactics can carry one or more decoys, such as a meaningful name for the database, mixing non-sensitive original data within the bogus data, and enticement data within the database that points to other fake resources in the system.

Deception stories are described in terms of scenarios and intended biases exploitation. We borrowed the Given/When/Then structure used by the Test-Driven Development community to specify deception stories as executable specifications [44]. In this structure, the ‘given’ part describes the state of the world before you specify a behavior, i.e., it represents the pre-conditions to hold the story. The ‘when’ section is the specification of the behavior. Finally the ‘then’ section describes the changes you expect due to the specified behavior. Listing 1 shows the first scenario specification, where the attacker reaches the log file. The expected outcome of this scenario is an alert sent to a security management system to indicate an attempt to access the bogus database. We describe the story, biases exploitation and associated deception tactics as comments within the specification structure to keep stories in a simple format.

Listing 1: Deception story where attacker reaches the log file

```plaintext
Given
the system has some vulnerability
And
an attacker gains access to the database using bogus user accounts
And
the attacker finds the log file
And
the attacker gains access to the database using the collected passwords
When
the attacker tries to gain access to the database
Then
an alert is sent to a security management system
```

The second story (Listing 2) shows a specification where an attacker gains access to the bogus database using the collected user and guessing the password by using brute force attack. As the password is fabricated to be vulnerable, the authentication is supposed to be easily overcome, keeping a plausible story. After the execution of the authentication, bogus databases are available to be connected.

Listing 2: Deception Story where attacker gains access to the database

```plaintext
Given
an attacker guesses the password of the bogus account (brute force attack)
When
the attacker authenticates in the DBMS
Then
an alert is sent to a security management system
```

The third story (Listing 3) encompasses the detection of DDL commands sent by the attacker to the bogus database. Monitoring these activities assists to give more plausible stories to the attacker. For example, in this story, the attacker is presented with an excuse that DDL commands are not allowed by using this user.

Listing 3: Deception story where attacker sends DDL commands to a database

```plaintext
Given
DS2
```

Table I: Deception Strategy Properties

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Database access deception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals</td>
<td>a) Identify attackers trying to gain spurious access to the smartphone file system, b) Divert attackers with bogus data pointing to multiple directions</td>
</tr>
<tr>
<td>Monitoring Channels</td>
<td>a) Authentication function, b) Commands sent to the database server</td>
</tr>
<tr>
<td>Metrics</td>
<td>#Wrong authentication attempts, #Valid User Access, #DML Commands, #DDL Commands</td>
</tr>
<tr>
<td>Risks</td>
<td>a) Attacker identifies data as fake, b) Sensitive data is loaded as fake data, c) Fake database gives access to real database</td>
</tr>
</tbody>
</table>

Table II: Deception Tactics Properties

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Decoy</th>
<th>Technique(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1-Fake Database</td>
<td>Inventing</td>
<td>Repackaging</td>
</tr>
<tr>
<td>T2-Fake Database Tables</td>
<td>Simulate Original Structure</td>
<td>Repackaging, Inventing</td>
</tr>
<tr>
<td>T3-Fake Data Items</td>
<td>Mixed original non-sensitive data</td>
<td>Repackaging, Inventing</td>
</tr>
<tr>
<td>T4-Log Instrumentation</td>
<td>Database URL/User</td>
<td>Dazzling</td>
</tr>
</tbody>
</table>
On the other hand, DML commands should result in bogus data that lead attacker astray. This is specified in the story DS4 (Listing 4). Given that deception story DS2 occurs, whenever the attacker sends a DML command, the system will respond with bogus data. The content of these answers may be just random data or fabricated data with the intention of giving new directions of exploitation to the attacker. In this sense, new stories can be created to indicate new tactics and new biases exploitation.

Listing 4: Deception story where attacker sends DML commands to a database

```verbatim
/* Deception Story [DS4]: The attacker sends DML commands to bogus database
Biases exploitation: Confirmation Biases (look for data that confirm there is a real database), Personal bias (systems do not lie)
Associated deception tactics: T2, T3 */

Given DS2
When the attacker opens a bogus database
And the attacker sends a DML command
Then an alert is sent to management system
And the system responds with ‘unauthorized command’ to the attacker
```

Note that to assist forensic procedures, a unique bogus user should be instrumented on the log file for each employee’s smartphone; thus, database exploitation can be mapped to a unique device. Of course, exploiting the deceptive database does not necessary indicates that the smartphone file system was compromised. However, it represents a sign for further investigation, that would otherwise go unnoticed.

Multiple strategies can be built to compound a repertory of strategies that will be used during adaptation. We represent this repertory using a feature model describing variabilities, commonalities and constraints between deception strategies. For example, a strategy that is mandatory in the system or depends of another strategy to be executed is specified in the feature model. Figure 4 illustrates an example of a deception repertory containing three strategies. Under database access deception strategy, fake database tactic is considered mandatory, while others are optional. However, a dependency constraint sets that Log Instrumentation, when activated, demands the fake database tactic activated as well. Similarly, Fake database items depends on Fake database tables tactics active. Other strategies (2 and 3) are exemplified in the feature model containing other tactics. During adaptation, activation is possible on the strategic or tactical levels, assuming the constraints imposed by the engineer in the feature model. Due to lack of space, we will not cover adaptation policy specification. However, there are numerous mechanism that can support our approach.

VII. RELATED WORK

Many works aiming at proving a framework for planning deception-based defense were developed. Differently from our work, these frameworks provide too general activities that rarely form a practical structure to build deception strategies considering coordination and adaptation. Follow, we summarize some of the related works, emphasizing those that provide some specification (planning) model for deception-based defense. [17] presents a deception process as a planning loop that begins with the development of a desired deception goal in support of a strategic goal, followed by decisions (characteristics, potential stratagem, illusion, channel, and ruse), and then execution, where the deception is compared against its goals. [50] describes "See-Think-Do", a three step deception process. In this model, the planner deems what the target (attacker) sees from the deception operation, what the target should conclude to achieve the deception goal, and what the target should do as a result of the conclusions based upon the observations. [51] proposes a set of requirements for creating deceptive content, namely enticing, realistic, minimize disruption, adaptive, scalable protective coverage, minimize sensitive artifacts and copyright infringement, and contain no distinguishable characteristics. More recently, [52] presents a model to plan and integrate deception in computer security compounded by six steps: specify strategic goal of deception, specify the target reaction, identify attacker biases exploitation, identify the deception technique (simulation/dissimulation), specify feedback channels, identify risks and countermeasures. Additionally, the model describes activities for implementing and integrating deceptive components, and monitoring feedback channels. [53] presents a model based on iterations to help organizations to assess risks and effectiveness of denial and deception (D&D) operations. The model is a spiral model that considers four dimension for rapid prototyping and tuning of D&D techniques and services based on outcomes observed. The first dimension refers to the planning of cyber D&D. The second dimension implements the D&D using tools, threat data, shared repositories and metrics. The third dimension is devoted to deployment and execution, and the last dimension is about analysis and improvements.

VIII. CONCLUSIONS AND FUTURE WORK

This paper discusses some initial thoughts on how to integrate deception strategies in an holistic, adaptive and
evolutionary software design process, offering a set of activities, a conceptual model relating the fundamental concepts, a vision of an adaptive deception platform, and examples of stories illustrating attackers’ course of interactions. As part of our future work, we intend to develop a framework for deception specification using goal-oriented methodologies and the design and implementation of an adaptive infrastructure to support our approach.

ACKNOWLEDGMENTS

The first author is assisted by CAPES foundation under process number 0553-14-0. The authors are grateful for the support given by FCT and NOVA LINCS Research Laboratory (Ref. UID/CEC/04516/2013) during the development of this work.

REFERENCES