

# Stressing Security Requirements: Exploiting the Flaw Hypothesis Method with Deviatonal Techniques

Thitima Srivatanakul\* John A. Clark Fiona Polack  
Department of Computer Science, University of York,  
Heslington, York, YO10 5DD, UK.  
[jill,jac,fiona]@cs.york.ac.uk

## Abstract

*The Flaw Hypothesis Method has been widely used in the security community to ‘stress’ test system security. However, the approach of flaw generation to date only identifies threats that are already known for the domain. Deviatonal techniques are one approach to improve the process of the flaw hypothesis generation. The approach systematically perturbs elements of the system. In this paper we explore a generalised technique illustrated with example applications. The worked examples show that the technique forces more rigorous consideration of the security aspects of the system.*

**Keywords:** *Flaw hypothesis method, penetration testing, deviation, security, testing*

## 1 Flaw Hypothesis Method

The Flaw Hypothesis Method (FHM), first proposed by Weissman in 1973 [10], is one form of penetration testing and has been very influential. FHM has been mainly used as a tool for software certification. It uncovers weaknesses or flaws by simulating the attacks on the system, thus providing a method for assessing vulnerabilities. It can also help to identify future system security requirements. FHM aims to provide a comprehensive penetration testing technique and consists of four stages [10].

Typically, FHM makes use of documentation and other knowledge of the system to generate flaws or vulnerability hypotheses (stage 1). Once the generation of hypothetical flaws is complete, the flaws are confirmed (if assessed as true), or refuted (if assessed as false) by live testing or analysis of documentation or code (stage 2). The confirmed flaw instances are then generalised

(stage 3), to find other instances of the weakness and gain new insight. Known flaws are then repaired by fixing coding errors, or providing recommendations on the countermeasures needed (stage 4).

The effectiveness of FHM relies heavily on the flaw hypothesis process. However, there exist many problems and obstacles to achieving comprehensive flaw generation. This is partly because the process of flaw generation depends on the analyst’s skill, experience and familiarity with the system. It is not surprising that common vulnerabilities are still overlooked, since there is usually no systematic procedure on how flaw generation should be carried out. It is necessary that subtlety be included to determine what security for a system should comprise and how assurance can be given for the developed system.

Another hindrance to comprehensive flaw generation is the over-reliance on existing or documented vulnerabilities. Nowadays, there are many vulnerability taxonomies and databases and most often these are used for the flaw hypothesis. Since emerging types of system exhibit novel vulnerabilities, these collections prove inadequate.

Below we describe a general technique, based on deviation, that allows analysts to use their knowledge in a systematic and structured way and yet provide a framework for the imaginative anticipation of flaws.

## 2 Deviatonal techniques

In developing systematic approaches to security, we have explored techniques from other domains. In the safety domain, there are well-established approaches, based on the concept of deviations, that systematically identify hazards, their causes and consequences. Deviatonal techniques typically hypothesise that some aspects of a system differs from normal in some way and the analyst is invited to consider the perturbed system

---

\*Thitima Srivatanakul is funded by the Royal Thai Government.

in more detail.

Software fault injection simulates anomalies by instrumenting a piece of code, then executes the code to observe the impact of anomalies. One particular form of the software fault injection, mutation testing, is used to assess how thoroughly a test set exercises a program. HAZOP [5] applies deviations to design representations. The identification of deviations is prompted by a set of guide words. The generic list of guide words include: *no, more, less, as well as, more than, part of, reverse, other than, early, late, before, after*.

Deviation techniques have a long-standing record of successful use in the safety domain. They are used because they force analysts to consider deviations that have not been considered. The investigation of unintended behaviours or unexpected situations, prompting unknown threats and vulnerability, is also essential in the security domain. This paper seeks to show that such techniques can be adapted for use in security, based on the Flaw Hypothesis Method framework.

### 3 Example Applications

This section illustrates three instances of the generalised approach. The worked examples are at different levels of formality and relate to different phases of development.

#### 3.1 Flaw hypothesis on use case model

The first application example applies the deviational approach of HAZOP, tailored to a security perspective, to a use case model [2]. We provide more detail in [7].

The approach systematically mutates the model and its elements, thus prompting consideration of potential flaws in the system. The elements subject to deviations are the actor's attributes (intent and capability), associations of actors and the use case, and use case elements (pre-condition, sequence of actions and post-conditions). An apparently-innocuous web ordering system provides an example application. Here, we illustrate deviations from the ordering process ('order goods' use case) of the system with the 'customer' as the associated actor.

**Deviations from actor's intent** - Deviations from intent (deviations from goals), whether intentional or accidental, can reveal potential threats from malicious actors, by considering deviation from expected actor behaviour. In our example, the customer's intent is to order goods. Applying, for example, the guide word 'more' to the customer's intent, we derive, 'the customer orders excessive amounts of goods', exposing hy-

pothesised flaws in the ordering process. The potential causes for this deviated intent are then investigated.

**Deviations from association** - This part of the analysis hypothesises flaws concerning the restriction of operation access to a particular group of users (an actor), and of assignment of access controls to a particular user or actor. For example, here, applying the guide word 'more' prompts the analyst to consider the causes and effects of having multiple ordering sessions from one customer. Deviation analysis must also consider the possibility of unexpected interactions through shared and multiple roles. For example, having the same person to both process and approve a payment should not be allowed.

**Deviations from use case description** - Deviations from the normally-reached states of pre- and post-conditions may have an adverse effect on security. The analysis of the use case must elicit possible causes of such deviant states. We can also address the variations from the normal behaviours (use case actions), by considering the deviations of each step in the use case and investigating the causes.

In the 'order goods' use case, one of the use case actions is 'the customer submits payment detail'. The description can be deviated using 'other than' guide word, for example, forming 'an attacker submits payment details'. This event could occur by an intruder modifying the message, for example.

The above examples use HAZOP to generate hypothesised flaws in a systematic way. These flaws could represent alternative behaviours or abuses (c/f abuse case model [4]) in a system. HAZOP can also provide a systematic approach to reasoning about the high-level security of a system modelled in use cases, and can highlight issues on lower level design possibilities. It is demonstrated in this example application [7] that flaw generation process can be integrated with the system development and that potential flaws and security issues can be considered at the start of system development.

#### 3.2 Flaw hypothesis on zonal requirements

In security, there are many examples of 'zones'. In airports, for example, there are many physical zones and access between them should be strictly controlled. Zones can also be defined for IT systems. For example, information may be compartmentalised with restrictions on who may access what and how.

In designing systems, it is intended that the channels used between these zones are the only relevant ones. Physics often implements unintended opportunities for associations. Covert channels can be considered

as unintended associations. This section explains the technique of security zonal analysis, and provides an example application on the deviations on information flows between zones, in order to consider hypothesised flaws from unintended associations between zones. The approach is extracted from [8].

The method starts by exploring the scope and model of the system. We consider the security implications of possible interactions of the target zone (the zone in which the security properties will be analysed) with other zones. These zones are identified by considering their relationships with the target zone. The possible relationships between zones are, for example, physical proximity, functional dependency, temporally related or procedurally adjacent.

The vital part of the process is to be able to identify properties of the boundary between zones and what flows between them (this can be physical-material flows, energy flows, logical flows through physical, temporal or procedural channels). These are then subject to deviational analysis as before.

For each identified property, HAZOP guide words are applied to identify possible channels through the boundary. The guide words used are generic, and with suitable interpretation may find useful application across a range of situations.

Unintended channels may be a major cause for concern, but one should consider also unfortunate effects of normal intended channels. It is, for example, possible for insiders to abuse the privileges that they have. The causes of any unfortunate interactions should be investigated. The steps are repeated until all the channels and zones have all been considered.

As an example for the approach, we consider an office in our Computer Science Department, where access to resources in the office is restricted. Wall, LAN and electric cable are identified as interfaces/boundaries between the target office and an adjacent office. As an example, if LAN has ‘less’ freedom from corruption (one of its property), we could hypothesise flaws that can cause LAN cable to be exposed, making it easier to access. Recommendations or comments on barriers can be recorded for each identified cause.

Zones identified in our example [8] are mainly physical. The results illustrate the importance of physical zones as part of the analysis of system security, even when the system itself is rather simple. Physical zones are more important for the computing elements of system security than they have traditionally been (e.g. concerns have arisen recently in a more commercial setting due to the rise of smartcards, with crucial cryptographic key information being leaked via timing and power consumption behaviours). Never-

theless, the logical side of zonal analysis, addressing non-confidentiality aspects clearly need to be addressed (confidentiality is the subject of the Shared Resource Matrix Methodology [3], for example).

The application of deviational techniques to zonal properties shows that a channel can be abused in many ways. This represents potential flaws in the system. A perfectly legitimate channel can be exploited by unexpected interactions. Considering temporally related channels in zonal analysis revealed ‘time-of-check-to-time-of-use’ (TOCTTOU) problems. A thief might enter unimpeded during office hours and wait until the building is locked, when theft opportunities arise. All hypothesised flaws, if can be exploited, are confirmed true. Further recommendations to manage the risks are considered.

The analysis helps make explicit the purpose of the intended channels, highlights some of the unintended channels and identifies persons involved. It can also be used as evidence in an overall security argument that all the possible access to a zone has been considered. By explicitly focusing on all possible exploitation through other zones, additional elements of a security argument can be realised.

### 3.3 Flaw hypothesis on formal elements

This case study explains the deviation of a formal specification, using a CSP [1] security example. It also shows that the flaw generation process can be automated with a mutation tool. Further details are given in [9].

Formal development notations, such as Z, B and CSP, are used for high integrity systems because they introduce the possibility of: precise specification of requirements; formal (or highly rigorous) proof that a specification has particular properties or that a more detailed design refines the specification. In practice we need to consider the operation of the system under normal conditions and also under various failure conditions.

The case study applied specification mutation to bring about issues on specification validation and the effects of failure. The system consists of a number of smartcard-based electronic purses that carry financial value. The purses interact with each other to exchange value. The system possesses two important security properties (‘no value created’ and ‘all value accounted for’) which the system needs to maintain.

Two specifications are written to capture functional behaviour of the system and the security properties of the system. The functional specification must refine to the security properties specified. The approach mu-

tates the functional specification. The mutants are generated by a mutation tool [6], which parses the specification and injects mutants (faults) into it, according to a user-selected subset of the predefined mutation operators (modification rules). The mutated specifications are then checked for whether the security properties still hold or not. FDR2, a CSP model checker, is used to check whether the mutants maintain the security properties (in this case, it shows whether the mutants are still the refinements of the property specification).

If a mutant refines the secure property specification, it is said to be *equivalent*: it possesses the same properties as the original specification. If the mutant invalidates the security property, it can represent hypothetical failures due to failures of the execution environment or those engineered by an attacker. Here is a part of our case study specification.

$$\left| \begin{array}{l} \textit{FromPurse}(id, bal, lost, val) = \\ \quad \text{if } val \leq bal \text{ then } ( \\ \quad \quad \textit{ok} \rightarrow \textit{Purse}(id, bal - val, lost) \\ \quad \quad \sqcap \textit{lose} \rightarrow \textit{Purse}(id, bal - val, lost + val) \\ \quad \quad \sqcap \textit{ignore} \rightarrow \textit{Purse}(id, bal, lost) \\ \quad ) \text{ else } (\textit{ignore} \rightarrow \textit{Purse}(id, bal, lost)) \end{array} \right.$$

It is the CSP specification for the process *FromPurse* (or the paying purse). The purse can non-deterministically choose ( $\sqcap$ ) to *ok* the transfer, which decrements the balance value; *lose* the transfer, which decrements and increments the same value to the balance and lost components; or *ignore* the transfer, which does nothing.

The mutation operator used here is an ‘event replacement’ operator, where an event is replaced with another event. The mutant replaces event *ok* with event *ignore* in the process that acts as the *FromPurse*, causing the balance component to change when engaging in ignore.

The mutant allows the paying purse to decrement the balance while engaging in *ignore*, which means that no other purse increments the balance, so this violates the property, ‘all value accounted for’. Once we know that this behaviour violates the security property, we perform further investigation of how this might be engineered by an attack on an implementation of the specification. This is where the hypothesised flaws are explored whether they are assessed as true or false. The identification of flaws that break the security policy can focus verification effort on the implementation and test case production.

We find that, in contrast with software testing, equivalent mutants are valuable for specification validation. Specification mutation can expose the fragility

of verification evidence to specific choices of parameters. The results may show that the evidence does not generalise from specific successful instances or else increase confidence that it does so. Specification mutation can be used to challenge various assumptions and in doing so complement the areas where formal methods currently succeed. We believe that specification mutation has much to offer.

## 4 Conclusion

This paper describes a generalised technique for the flaw hypothesis process of FHM and extends its use to security requirements analysis within the development process. The deviation concept is used as the basis for our approach, which is explored with some worked examples for system models at different levels of formality. We regard our approach as a complement to existing approaches of flaw generation.

## References

- [1] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
- [2] I. Jacobson and M. Christerson and P. Jonsson and G. Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.
- [3] R.A. Kemmerer. The shared resource methodology: An approach to identifying storage and timing channels. In *ACM Trans. of Computer Systems*, pages 256–277, 1983.
- [4] J. McDermott. Using abuse case models for security requirement analysis. In *Proc. of 15th Annual Computer Security Applications Conference*, 1999.
- [5] F. Redmill and M. Chudleigh and J. Catmur. *System Safety: HAZOP and Software HAZOP*. John Wiley & Sons, 1999.
- [6] T. Srivatanakul. Mutation testing for concurrency. Master’s thesis, Department of Computer Science, University of York, UK, 2001.
- [7] T. Srivatanakul and J. Clark and F. Polack. Effective security requirements analysis: Hazop and use cases. In *Proc. of the 7th Information Security Conference*, pages 416–427, 2004.
- [8] T. Srivatanakul and J. Clark and F. Polack. Security Zonal Analysis. Technical report, YCS-2004-374. Dept. of Computer Science, University of York, 2004.
- [9] T. Srivatanakul and J. Clark and S. Stepney and F. Polack. Challenging Formal Specifications by Mutation: a CSP security example. In *Proc. of the 10th Asia-Pacific Software Engineering Conference*, pages 340–350. IEEE, 2003.
- [10] C. Weissman. System Security Analysis/Certification Methodology and Results. Technical report, SDC SP-3728, 1973.