

# Security vulnerabilities detection using model inference for applications and security protocols

## Position statement

K. Hossen\*, R. Groz and J.L. Richier

Université de Grenoble, LIG lab  
F-38402 St Martin d'Hères Cedex, France  
{hossen,groz,richier}@imag.fr

**Abstract**—“Internet of Services” (IoS) is a vision of the Internet of the Future where applications are built by combining services provided by a variety of service providers over the network. They are deployed as needed and consumed at run-time in a demand-driven and flexible way. Model-based testing is one method for testing security of applications but it needs formal models and most of the time service providers are not able to provide them. For that, model inference methods adapted to security testing can be used. This document tries to give some directions in order to combine enhanced model inference and model testing to ensure security of services automatically.

**Keywords:** *model inference, model-based testing, security*

### MODEL INFERENCE

#### *In the context of the composition of services*

In IoS (Internet of services) vision, we are considering large applications built by assembling components provided by various service providers. In this context, validating security properties is difficult as standard tools may not be applicable. Due to the complexity and the fact that most of the time service providers are unable to provide a formal model of their applications, model-checking or model-based testing turns out to be impossible or inefficient. Even if we focus on specific components, the absence of model to analyse their interactions with other services can invalidate property verification.

However, in a testing context, partial models can actually be retrieved from observations gathered during the testing process. This statement still holds for black box components. Model inference – that is, methods to automatically derive models from observation of the behaviours – can facilitate testing of services by automatically providing models of the components and with a variable level of details and information, depending on the goal of testing (checking the behaviour of an application, ensuring security properties...), but also on the time and computational resources available.

Model inference is commonly divided into two categories: active learning and passive learning. Passive learning gathers knowledge only from a given set of

observations whereas active learning is free to interact with the tested system. In our case, all services have an open interface that is why active learning, which converges faster and has better results than passive learning, is the best choice.

Angluin's well-known algorithm (L\*) [1] is an example of the active approach and it can learn a deterministic finite automaton (DFA) with a number of polynomial tests regarding the size of the automaton, assuming an oracle provides counterexamples for incorrect models. This algorithm is commonly used as a basis for developing enhanced learning algorithms. But other algorithms may be more adapted to model services with I/O parameters [2].

#### *Enhanced Model inference*

In IoS vision, web-based applications massively use structured messages to communicate. As a consequence, the behaviour of applications is mostly characterized by its inputs and outputs (I/O) and not only by its states. DFA modelling by flattening I/O space into a single alphabet would be highly inefficient because the number of states in the machine tends to explode trying to represent each I/O possibility by a different state. In order to correctly use models, we must have an efficient model that natively considers I/O and structured inputs and outputs.

Various methods have already been proposed to enhance Angluin's algorithm to infer a Finite State Machine (FSM) with inputs and outputs, while reducing the cost of converting from Mealy to Moore machines. Typically, [3] changes the structure of the observation table to record both inputs strings and outputs strings. In [4], Neise defines filters to reduce the observation table.

In [5] and [2], the model was enhanced again to consider input/output parameters with finite (or infinite) domains; this is called Parameterized Finite State Machine (PFSM). A different extension to infinite parameter domains was proposed by [6]. An Extended-FSM (EFSM) is also proposed in [7]; its main difference with PFSM - when inferring black-box is considered - is that the last one does not use variables because they are not distinguishable from

---

\* Work funded by project SPaCloS (n°257876, FP7-ICT-2009-5, ICT-2009.1.4: Trustworthy ICT)

states in the internal structure. Such models have clearly been designed with security modelling in mind. More specifically, in [8], Shu and Lee defined a symbolic parameterized finite state machine (SP-FSM) to model security protocols. They also defined a representation of a security message containing Integer, Key, Nonce (random number used once) and Constant.

#### PROPOSED DIRECTIONS

##### *Modelling services for security testing*

Angluin-based algorithms are designed for cases where we know the entire input vocabulary. Basically, we select a state and for each element of the vocabulary we ask the oracle whether there is another state, or not. This method provides a simple way to know when to stop the algorithm but doing all the tests can take a lot of time due to the number of states and inputs in the system.

Furthermore, when concentrating on the validation of security protocols as considered by [8] and [9], the problem is more complicated because we have to take into account security properties and find a way to relate abstract and concrete parameters, in the presence of complex data structures (typically in XML). These properties and structures need to be modelled before model-based testing and relevant levels of abstraction have to be defined.

This has been studied in AVANTSSAR project where languages have been developed to that aim such as an extended version of HLPSSL [10], Aslan and Aslan++ [11]. When it comes to testing services in a production environment, we must consider the intricate relation between functional and security requirements because services in the environment may impact the security of an application even though they may not take part in a cryptographic protocol (DDoS attacks can entail the availability of critical services and potentially create security violations).

State-based models with variables, e.g. HLPSSL or Aslan, provide a framework that takes into account management of security related parameters (such as keys, nonces and encrypted messages) as well as functional behaviour. Most modelling languages, which are based on Dolev-Yao representations of crypto-protocols, consider abstract representation of data, which is suitable for protocol verification. However, when it comes to testing actual systems, mapping between abstract representations and complex input/output parameters and data structures is a non-trivial task. The formalization of mapping structures for learning of state based models has been investigated in [7].

As presented in [5], Angluin-style learning can also be considered as a restricted type of the Vasilevski-Chow [12] approach to FSM testing and identification, where the characterization set is approximated to sequences extracted from counter-examples provided by the oracle. Thus it is also a form of systematic model driven test derivation technique,

which could complement other model-based testing approaches.

##### *Model refinement by testing*

For adequate testing, information on some parameters must be kept, which implies some kind of enhanced state representations with possibly non-finite domains and restricted relations as in [9], and the relation between input and output parameters must be addressed. This can be captured in models such as PFSSM [2], but the inference may use other techniques for input/output inference, based on invariant inference [13] with a tool like Daikon [14] or statistical clustering [15].

A key issue in model inference is model refinement. In the case of Angluin's based approach, the refinement is done by testing each state using each element of the vocabulary and then asking an oracle if the model is correct or if a counterexample exists. This counterexample is re-injected in the model inference process to refine the models. This approach needs to collaborate actively with all testing processes available and a deterministic oracle, which, considering the cardinality of inputs of typical services may not be realistic.

When a model is available, most model-based testing approaches will either target specific structures, or combine requirements and models to derive tests that could lead to violation of required properties. The results of such tests can be used in a passive learning approach to derive models at a lower level, or to refine existing models. Since testing is done at a lower level (with parameter details), such models are actually described at a more detailed level than the initial ones. In our active learning framework, the tests results could also be added to the observations, and would trigger further test derivation to complete the observation tables or structures.

Two additional approaches for testing system security violations and reliability have been suggested in [5]:

- To submit specially crafted invalid inputs to the models and then observe the behaviour of the machine and especially invalid outputs, states reached (for filter-based security). Such inputs are concretizations of abstract inputs considered in the learning alphabet.
- Fuzzing from learned states. By injecting invalid, malformed or random sequences as inputs at each state of the automata, fuzzing is a simple way to test a lot of states efficiently and this approach can be viewed as a particular implementation of oracle to provide counterexamples [8].

Actually in the classical approach, fuzzing gives us only boolean information, e.g. crash/error or not, but along with learned models, it can detect additional states much earlier in the process. Accurate harnessing and a robust abstraction/concretization implementation of inputs and outputs are needed.

Refinement can be improved using attacker models linked to a library of vulnerabilities. Attacker can target specific states and discover error using input sequences. State distinguishability, needed for inference, can be based on sets of input sequences, as in [16]. In order to be more realistic and SOA adapted, classical models such as introduced by Dolev & Yao have to be extended. In [17], some extensions are defined to address the following attacks:

- Malformed inputs: Specially crafted XML messages are used by components to communicate. However, the standard Dolev-Yao model is insufficient to address all the problems. In [18], Backes and Groß give some examples of problem. The extended model should consider the fact that ordering of nodes often does not matter and malformed input have to be rejected or ignored safely
- Guessing attacks [19]: The attackers use stored messages (off-line approach) to guess important information (impossible with Dolev-Yao) if it has low cardinality, e.g. passwords containing only digits and shorter than 8 characters, and a small entropy, e.g. problem discovered in 2008 in the OpenSSL package of Debian.

Of course, more security aspects should be explored and case studies will be of great help to find interesting attack sequences. Furthermore, the attacker itself could be extended to consider, for example, multiple non-collaborating attackers and their interactions as done in [17].

These more realistic attackers will be able to use vulnerability databases to select tests that are relevant for the case, to perform these tests and refine the selection, depending on the results, and to update the vulnerability library with attack sequences that would be found.

Vulnerability library provides complete vulnerability models and documentation. The SHIELD project provides an example of such a library in [20]; the Security Vulnerabilities Repository Service (SVRS) handles the management of vulnerability models, e.g. up/downloading and searching by multiple means.

#### CONCLUSION

This document seeks to lay the basic principles which we are exploring in order to combine in an efficient and relevant way model inference and model-based security testing. Enhanced models associated with an adapted formal language for security purposes, along with abstract models for attack and their associated concretizations are key ingredients for inferring useful models for security testing.

#### REFERENCES

- [1] D. Angluin, "Learning Regular Sets from Queries and Counterexamples", *Information and Computation*, 75, 1987.
- [2] K. Li, R. Groz, and M. Shahbaz, "Integration Testing of Distributed Components based on Learning Parameterized I/O Models", In FORTE 2006, LNCS 4229, pp. 436-450, Paris, France, 2006.
- [3] M. Shahbaz, R. Groz, "Inferring Mealy Machines", *Formal Methods*, pp. 207-222, Eindhoven, the Netherlands, 2009.
- [4] O. Neise. "An Integrated Approach to Testing Complex Systems", Ph.D. Thesis, University of Dortmund, 2003.
- [5] M. Shahbaz, "Reverse Engineering Enhanced State Models of Black Box Software Components to Support Integration Testing", Ph.D. thesis, Institut Polytechnique de Grenoble, 2008.
- [6] Berg, Jonsson, Raffelt, "Regular Inference for State Machines Using Domains with Equality Tests", in FASE 2008, LNCS 4961, 2008.
- [7] Fides Aarts, Bengt Jonsson, and Johan Uijen, "Generating Models of Infinite-State Communication Protocols using Regular Inference with Abstraction", *ICTSS*, Natal 2010.
- [8] G. Shu, D. Lee, "Testing Security Properties of Protocol Implementations - a Machine Learning Based Approach", *ICDCS*, Toronto, Ontario, Canada, 2007.
- [9] Armando, R. Carbone, L. Compagna, K. Li, G. Pellegrino, "Model-checking Driven Security Testing of Web-based Applications", *MDV Workshop*, collocated with ICST, Paris, 2010.
- [10] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Modersheim, and L. Vigneron, "A High Level Protocol Specification Language for Industrial Security - Sensitive Protocols", in *Proc. SAPS'04*. Austrian Computer Society, 2004.
- [11] AVANTSSAR. Deliverable 2.2: ASLan v.2 with static service and policy composition and 2.3: ASLan final version with dynamic service and policy composition. Available at <http://www.avantssar.eu>, 2008.
- [12] M. P. Vasilevski. *Failure diagnosis of automata*. Cybernetics and Systems Analysis, 1973.
- [13] M. Shahbaz, R. Groz, "Using Invariant Detection Mechanism in Black Box Inference", *ISoLA Workshop on Leveraging Applications of Formal Methods*, Poitiers, December 2007.
- [14] M.D. Ernst, J.H. Perkins, P.J. Guo, S. McCamant, C. Pacheco, M.S. Tschantz, and C. Xiao, "The Daikon system for dynamic detection of likely invariants", *Sci. Computer Programming* 69, pp. 1-3, 2007.
- [15] Arnaud Dury, Hesham Hallal, Alexandre Petrenko; "Inferring Behavioural Models from Traces of Business Applications", *ICWS 2009*: 791-798
- [16] Roland Groz, Keqin Li, Alexandre Petrenko, Muzammil Shahbaz: *Modular System Verification by Inference, Testing and Reachability Analysis*. *TestCom/FATES 2008*: 216-233.
- [17] AVANTSSAR. Deliverable 3.3: Attacker models. Available at <http://www.avantssar.eu>, 2008.
- [18] M. Backes and T. Gross. Tailoring the Dolev-Yao abstraction to web service realities. In *ACM Secure Web Services Workshop (SWS)*, pages 65-74, 2005.
- [19] G. Lowe. Analysing protocol subject to guessing attacks. *Journal of Computer Security*, 12(1):83-98, 2004.
- [20] SHIELD. Deliverable 3.1: Initial Repository Specification and Design. <http://shields-project.eu/>. 2009.