Learning to detect intrusion strategies

John Heasman and Steve Moyle

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, England

Abstract. Intrusion detection is the identification of potential breaches in computer security policy. The objective of an attacker is often to gain access to a system that they are not authorised to use. The attacker achieves this by exploiting a (known) software vulnerability by sending the system a particular input. Current intrusion detection systems examine input for syntactic signatures of known intrusions. Only checking for syntactic signatures presents a severe weakness in current intrusion detection technology, for it is relatively simple for an attacker to modify a known attack input such that it will avoid syntactic detection, while the attack is operationally semantically equivalent. That is the attacker gains access to the system by simple modifications to the known attack input.

A richer framework is required to consider the *operational semantics* of a potentially malicious input with respect to the context of the system. This work develops such a framework for a particular class of vulnerability using logic programming as the basis. The vulnerability studied was the common *buffer overflow*, an attack that exploits programming sloppiness and is based on the use of the operating system stack. This form of attack is acknowledged as being the most costly to enterprises world wide.

So far, a number of variations on stategies that exploit the buffer overflow condition have been determined in the real world. Some intrusion detection systems have implemented signatures for checking for these strategies. However, there are a vast range of alternative strategies that achieve precisely the same outcome – a successful attack. There are many common components in the strategies including *NOP Sledges*, *generating null bytes*, and *System Call execution*. This work develops a suitable logic programming basis for specifying the semantics of the components of attack strategies individually, and then assembling them to detect entire attacks. Furthermore, such logic programs can then be used as a means of detecting attacks in previously syntactically-unseen inputs.

The logic programming framework for buffer overflow intrusion detection was used as background knowledge in an emperical study for the induction of rules to detect attacks from examples using ILP. The background knowledge included a disassembler for the Intel x86 processor family, as well as predicates to define higher level attack functions in a particular context. The disassembler acts as a syntactic-to-operational semantic bridge, so that the higher level functions need only consider the semantics of the operations required. Experiments of learning ten different attack strategies to exploit the buffer overflow demonstrate that accurate theories can be generated from very few attack examples.