

Improving Signature Testing Through Dynamic Data Flow Analysis

Christopher Kruegel
Technical University Vienna
chris@auto.tuwien.ac.at

Davide Balzarotti, William Robertson, Giovanni Vigna
University of California, Santa Barbara
balzarot,wkr,vigna@cs.ucsb.edu

Abstract

The effectiveness and precision of network-based intrusion detection signatures can be evaluated either by direct analysis of the signatures (if they are available) or by using black-box testing (if the system is closed-source). Recently, several techniques have been proposed to generate test cases by automatically deriving variations (or mutations) of attacks. Even though these techniques have been useful in identifying “blind spots” in the signatures of closed-source, network-based intrusion detection systems, the generation of test cases is performed in a random, unguided fashion. The reason is that there is no information available about the signatures to be tested. As a result, identifying a test case that is able to evade detection is difficult.

In this paper, we propose a novel approach to drive the generation of test cases by using the information gathered by analyzing the dynamic behavior of the intrusion detection system. Our approach applies dynamic data flow analysis techniques to the intrusion detection system to identify which parts of a network stream are used to detect an attack and how these parts are matched by a signature. The result of our analysis is a set of constraints that is used to guide the black-box testing process, so that the mutations are applied to only those parts of the attack that are relevant for detection. By doing this, we are able to perform a more focused generation of the test cases and improve the process of identifying an attack variation that evades detection.

1. Introduction

Intrusion detection systems (IDSs) can be broadly divided into two classes: those that rely on models of normal behavior and detect deviations from these models (i.e., anomaly-based systems), and those that contain descriptions of malicious behavior and detect events (or sequences of events) that match these descriptions (i.e., signature-based systems). While both classes of intrusion detection

systems have complementary strengths, they are both vulnerable to evasion attacks.

In the case of anomaly-based systems, evasion techniques are used to craft an exploit so that it resembles normal behavior. The application of these techniques is usually called a *mimicry attack* [30]. In the case of signature-based systems, evasion techniques are used to modify an exploit so that it does not match any of the signatures used by the intrusion detection system, while retaining the ability to compromise the security of the target system [20].

Recently, a number of approaches [4, 13, 16, 18, 22, 23, 29] have been proposed to test the effectiveness and precision of network-based intrusion detection systems. In particular, approaches based on the generation of test cases by automatically deriving variations (or mutations) of known exploits have been shown to be able to identify problems in the detection mechanisms used by both open-source and commercial, state-of-the-art systems [13, 29]. These approaches leverage a number of transformations, called “mutant operators”, that are applied to an exploit template. The goal of applying these mutation operators is to obtain a modified version that has a different network manifestation with respect to the original attack, but it is still able to compromise a vulnerable target.

Mutant operators can work at different levels of abstraction (e.g., at the network level or at the application level), and they can be composed and/or applied multiple times. For example, consider a first mutant operator that adds effect-free commands to an FTP session (e.g., adds a “CWD .” or a “NOOP” command) and a second one that applies fragmentation to the IP traffic. The first operator can be applied multiple times to an FTP-based exploit template without invalidating the attack (unless, of course, the length of the session affects the success of the exploit), while the second one can be applied in different ways (e.g., by specifying different fragment sizes). Thus, the number of possible variations of the original exploit that can be used as test cases quickly grows very large.

In current approaches, the generation of test cases is either manually guided or a random process. In the former case, a human expert selects which operators to apply to the exploit template and which parameters to use for each

operator. The results obtained by running the selected test cases might provide hints on how to select the operators and their parameters in the next round of tests. In the latter case, the operators (and the values of their parameters) are selected randomly. Both these approaches are less than optimal because they either require extensive expert knowledge or represent “shots in the dark.” Therefore, there is the need for a new technique for testing network-based signatures that is both automated and more focused than a purely random approach. In theory, some guidance about how to generate the relevant test cases can be derived from the signatures themselves. For example, by looking at which features of the network traffic are analyzed by a signature, it is possible to focus the test case generation by using only the mutant operators that affect those features. Unfortunately, most intrusion detection system vendors do not make their signatures available because they consider them to be their intellectual property and an advantage with respect to their competitors. Thus, in general, one cannot rely on the availability of the signatures to guide the generation of the test cases.

To address this problem, we propose a novel approach to drive the generation of test cases based on the analysis of the dynamic behavior of a network-based intrusion detection system. As a first step, we apply dynamic data flow analysis techniques to the NIDS binary to determine which parts of the attack trace are checked by the NIDS. We then leverage this information to restrict the test case generation process to only use the mutant operators that modify the relevant parts of the attack.

Based on the knowledge of *which* parts of a network trace are considered by the detection process, we further refine our analysis to also take into account *how* these parts are used. For simple checks (e.g., the comparison of a source port number with an integer constant), the constant value specified by the signature is extracted from the dynamic trace. Most of the signatures also specify strings or regular expression to be matched against the packet payload. To address these cases, we developed a technique that aims at reconstructing a finite state machine that captures the behavior of the pattern matching process. That is, the state machine derived from the analysis should accept an input string if and only if this string matches a pattern specified by the signature. While it might not always be possible to precisely reconstruct this state machine (particularly in the case of regular expressions), patterns can be reconstructed by observing the execution of popular string matching algorithms such as Boyer-Moore [5] or Aho-Corasick [1].

The contributions of this paper are the following:

- We present a novel, practical technique to effectively drive the generation of test cases for the evaluation of network-based signatures. Our technique analyzes the dynamic behavior of a NIDS program to determine

which parts of an attack are used by the detection process.

- In addition to locating the parts of the attack traffic that are used in the detection process, we also determine the nature of the checks that the NIDS performs. In particular, our analysis can automatically extract both specific numerical values and strings that the NIDS is searching for.
- We have developed a prototype tool to evaluate our technique on both open-source and closed-source commercial NIDSs. The results demonstrate that our approach allows for effective generation of exploit mutants that are able to avoid detection, even when no signature information is available.

The remainder of the paper is structured as follows. Section 2 presents the dynamic analysis technique utilized to analyze the behavior of the IDS being tested. Section 3 introduces our mechanisms to extract signature constraints from the observed behavior. Section 4 explains how the analysis results can be used to generate the test cases for a network-based signature. Section 5 evaluates the effectiveness of our approach. Section 6 discusses related work. Finally, Section 7 draws conclusions and outlines future work.

2. Dynamic Data Flow Analysis

The goal of our dynamic data flow analysis is to determine which parts of a network stream are used by the intrusion detection system to identify the presence of an attack. More precisely, we are interested in the positions of all values, or bytes, that are analyzed by the IDS during the detection process.

To determine the input bytes that affect detection, we dynamically monitor the intrusion detection sensor while it is processing the network data. In particular, we tag each input byte that is introduced into the address space of the IDS process with a unique label. This label establishes a relationship between a particular input byte and a location in memory. Then, we keep track of each labeled value as the sensor’s execution progresses. To this end, the output of every instruction that uses a labeled value as input is tagged with the same label as well. For example, consider the case of a data transfer operation that loads a value with the label “123” from memory into a register. After the instruction is executed, the contents of the target register is also labeled with “123”. Clearly, it is possible that the result of an operation depends on more than one input byte. For example, consider an operation that adds together two values, each of which is tagged with a different label. In this case, the result is tagged with a set that holds both labels (called a *label set*).

