Application of Formal Language Theory in the IT Security area

Author: Lukáš Antal, iantal@fit.vutbr.cz

Security of complex systems is still an important issue these days. Every complex computing system consists of several components since the composition is the primary engineering method of complex system construction. Different components of the system communicate using protocols. The protocol in this context doesn't mean only network protocol but also file format specification, set of serialization rules and so on. Most security flaws exist because of these protocols are not formally specified. It is not an exception that in a one system, there exist mutually intelligible dialects which mean that components generally understand each other, but there are some ambiguities, underspecifications or implementation differences that could be misused by an attacker to exploit or even compromise whole system.

In 1980 Postel's (Robustness) law, which was formulated in the RFC 793, says "Be conservative in what you send, liberal in what you accept.". This law was formulated in the times when security was not the primary focus, but robustness of communication was what mattered. From today's perspective, this law is completely outdated. To be conservative in what you send is generally good for security, but author of the system component has to be also conservative in what to receive. In spite of Postel's law being obsolete, many systems, intentionally or not, still obey that rule and thereby increase attack surface.

In the last decade there were many cases of vulnerabilities in different protocols that emerged from the fact that rules of communication or message formats were not fully specified. It leads to the situation where different components implementing specific protocol implements unspecified or ambiguous pieces of protocol differently and thus allowing attacker to cause unexpected behavior. For example there was discovered a vulnerability in several implementations of RSA PKCS#1 allowing an attacker to forge valid RSA signature of arbitrary message. Different vulnerability in X.509 allowed an attacker to forge valid certificate to arbitrary CN (Common Name).

There are several different types of attacks caused by the situation described above but the presentation will be focused on Injection attacks. Injection attacks target applications at points where one system component acquires input from a user in order to construct an input for another component, such as a database, a scripting engine, or the DOM environment in a browser. The attacker crafts an input to the first component that results in the constructed input producing some computation in the second component that falls outside the scope of the operations the system designer intended the second component to perform. Injection attack can be also described as an attack that leverage a weak boundary between control and data channels - component usually expects data but attacker injects code that is executed by component.

SQL injection (SQLi) is the specific type of injection attack that aims for the backend database. SQLi is the most dangerous yet one of the most spread vulnerability of the web applications. Defense against SQLi and also other injection attacks consists in the input validation. This input validation is commonly realized by the regular expressions via methods known as whitelisting or blacklisting. These methods are proved to be prone to high rates of false positives and force a developer to choose between security and usability. In my presentation I will focus on the input validation aspects from the Formal Language Theory perspective. As SQL is Context Free language (can be described using CFG/BNF) it can't be validated/parsed using regular expressions. I will describe usage of the formal method called Differential Parse Tree. Differential Parse Trees can be used by developers as a means of input validation, but could be used by an attacker as an exploit development technique. Also connection between different classes of input formal languages and possibilities of its validation against protocol specification will be discussed.