

Autonomic Application Security Management

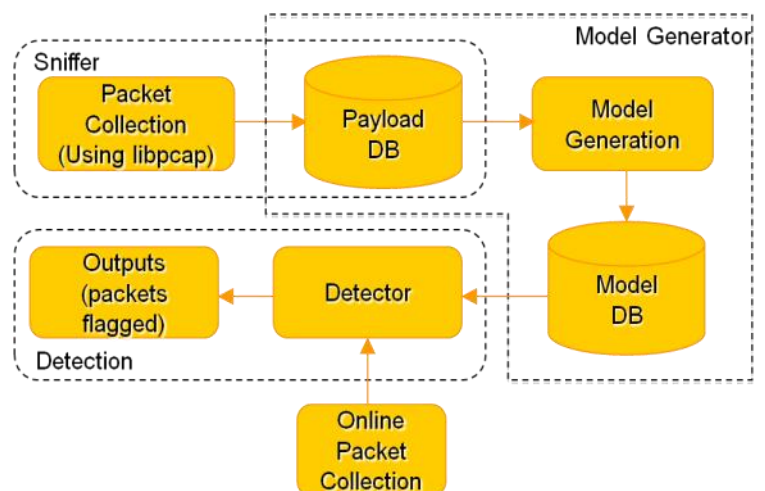
by Ram Prasad V

Network monitoring systems can be broadly classified into signature-based and anomaly-based. Signature-based systems are limited by the number of anomalies they can detect as they rely on signatures stored in a database) while anomaly-based systems have high false-positive rate. The existing payload/application anomaly detection systems use either byte distributions or work on the first line of the payload. Such an approach limits the number of attacks that can be detected and can work only for certain protocols (e.g., GET request of HTTP).

Our payload-based anomaly detection system—a major component of the Autonomic Network Defense (AND) System—classifies network traffic into objects such as headers, text, images, audio and video. The system consists of three major routines: (i) A sniffer module to collect normal traffic and store it in a database; (ii) A model generator routine which uses the collected traffic to model each of the above mentioned objects; and (iii) a detector routine to scan real-time traffic to detect deviations from the normal behavior.

We have implemented the following HTTP header models:

1. **Language model:** This model is used to profile the **byte distribution** of the HTTP headers. It helps us in detecting anomalies such as **shell code injection** as HTTP uses **ASCII based headers** and presence of code will alter the byte distributions present in the packets.
2. **Keyword/Value based model:** These models divide the headers into “keyword-value” pairs. The standard keywords are specified in the HTTP specification (**rfc-2616**). The keywords are profiled and the various statistics are generated for the values of these keywords. From these statistics, we build the following models:
 - ◇ **Keyword average, maximum and mode:** The mode, the average length and the maximum length of the **value** for each **keyword** are determined during the model building stage. During the detection phase, the profiled values are checked against the real time values and any deviation is flagged.
 - ◇ **New keywords:** HTTP allows users to define new keywords. But the interpretation of the keywords depends on the server and the client. Hence, the presence of any new keyword denotes a deviation from the normal behavior (especially since we profile the normal behavior of the traffic).
 - ◇ **Keyword ordering:** We study the ordering of keywords in the normal case. The ordering is profiled and any deviation from the normal ordering is flagged. The change in the ordering can be attributed to **hand crafted packets** or **buffer overflow** attacks.
 - ◇ **Time window based model:** Our detection routine works in a time window. The traffic is collected for certain duration, say **10 seconds** and then the models are verified. At the same time, the traffic is analyzed over the time window. Any similarity of packets is flagged as it can correspond to **scanning** or **denial of service** attacks.



We have developed a framework called AND - Tcpdump framework to monitor multiple application level protocols simultaneously. Based on which applications to monitor, the framework segregates the payload information and hands-over relevant packets to different application protocol handlers. The framework now has the capability to monitor the control port of FTP and detect all the data ports where the transfers are scheduled to happen.

The next stage of our project will be to profile java scripts present in text-based traffic. The script from the payload is extracted and byte codes are generated using the spider-monkey web engine of the Mozilla browser. The generated byte codes are then profiled. During detection, the generated byte codes are compared with the profiled values. We are also looking at analyzing image traffic. In case of image traffic, the analysis will look at the various distributions. Anomalies will represent the deviation in distribution. We intend to use MATLAB to analyze and examine the image traffic during the model building stages.