



Introduction

In basic script based attack systems, a single attack script is executed from start to end, until the script either succeeds or fails. When attacking a well defended system, relying only on a single attack script will obviously reduce the chances for success. It will for instance be easier for a defender to block the attack as it unfolds.

Thus, typical attack systems have a collection of attack scripts at their disposal, allowing a more diverse sequence of attacks to be made. Often, such systems execute attack scripts one-by-one until one of them succeeds. This means that in some cases, a number of scripts must be executed to their conclusion, before a weakness in the defense is found. In the figure below an example of this is shown.

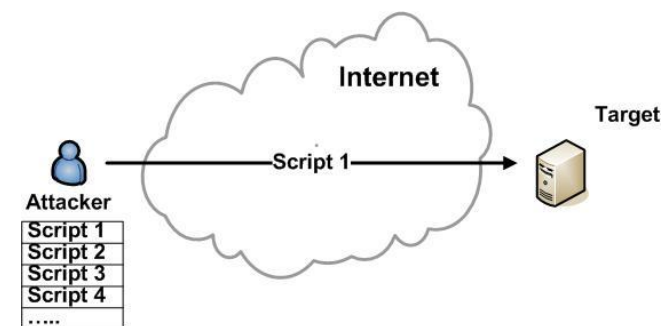


Figure 1: Script attack

Our aim was to exploit the weakest point in a defense as quickly as possible. The heart of our scheme is a system that starts executing all the available attack scripts concurrently, thus allocating resources uniformly. By evaluating the attack scripts as they unfold, however, the

system is to gradually concentrate resources on the most successful scripts.

By persistently exploiting the weakest parts of a defense in this manner, our goal was to breach it as quickly as possible.

Attack strategies

We have evaluated four different attack strategies in our thesis. Typical attack systems use a strategy where scripts are selected completely at random. There are two different approaches to this strategy:

Step Randomness - this strategy selects randomly between the different available scripts in each stage (every time a step in a script is attempted).

Script Randomness - instead of using all available script this strategy simply selects one of them at initialization.

To exploit the weakest point in a defense we invoked feedback based learning, resulting in two automata based strategies. Common for both of them are that the automaton will receive a reward if actions are completed successfully, else it will receive a penalty.

Linear Reward -ε-Penalty (LR-εP) - this automaton is based on an action probability vector. The parameters a and b control the impact of a reward and penalty. Probabilities for selecting each action are defined as $\pi_t(1)$, $\pi_t(2)$ and so on. Figure 2 and 3 show the formulas for estimating these probabilities when a reward or penalty is given.

$$\pi_{t+1}(d_t) = \pi_t(d_t) + \alpha [1 - \pi_t(d_t)]$$

Figure 2: Reward formula

$$\pi_{t+1}(d_t) = \pi_t(d_t) - b [1 - \pi_t(d_t)]$$

Figure 3: Penalty formula

Tsetlin automaton - this automaton is built up of states and state transitions. Each state corresponds to an action, and state transitions model the impact of rewards/penalties. The figure below shows an automaton that has two actions/scripts, and three states for each action.

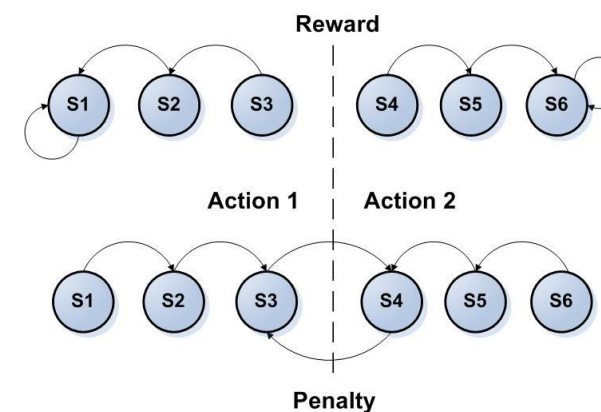


Figure 4: Tsetlin automaton

Results

We have developed a prototype which simulates two new attack strategies based on learning.

Our research is organized in terms of characteristic scenarios that highlight the main challenges that have to be dealt with. Defensive measures were gradually introduced, including patched for scripts and an active and dynamic defense system.

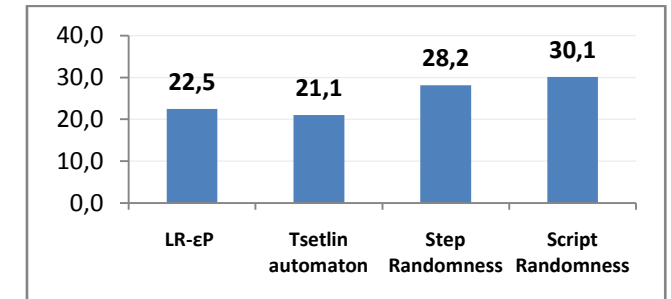


Figure 5: Average results for our strategies

In the figure above the average number of stages used to complete our scenarios is presented. Both of the strategies based on learning achieved on average significantly better results compared with the two completely random strategies. Tsetlin automaton was on average the quickest one.

The effectiveness of an intelligent system has been demonstrated to be depending on the environment the learning automata are operating in. However since the system's security is unknown to the attacker it is the average results that are representative.

The strategies were able to handle both of our defensive measures introduced at the target (patched for scripts and an active and dynamic defense). The results from our simulations indicate that the defensive measures were not detected. They were simply avoided when the strategies were experiencing lack of progress with the current selected script.

Conclusion

Our results demonstrate that a strategy based on intelligence would improve the performance of a system that tries to identify and exploit the weakest point in a defense.