Tunnel Hunter: Detecting Application-layer tunnels with statistical fingerprinting

In the paper “Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting”, Dusi et al. Propose Tunnel Hunter, a system for detecting tunneling activities over the HTTP and SSH protocols. Their motivation lies in the fact that tunneling activities are dangerous because they can disguise one application-layer protocol into another one and can make security policies ineffective and can lead to a dangerous illusion of security, therefore current security methods such as firewalls or Application-layer gateways are unable to detect them.

The authors start with an overview of tunneling techniques. They describe them as a client-server model, in which the client is a part of a protected network and the server is outside the network. If the client wants to start a connection with an external server by using a protocol which is prohibited by the network's firewall, the connection will be unsuccessful. By using tunneling, the client starts a connection with a proxy server, by disguising the packets of the prohibited protocol to packets of a non-prohibited one. Since the protocol is allowed, the connection is successful and the packets are sent to the proxy server, which forwards them to the actual destination server using the right protocol. The proxy sends packets to the client also by using tunneling. The authors mention that tunneling can be based on at least three protocols: DNS, HTTP and SSH.

The authors continue with some background on the term of statistical pattern recognition. They say that the goal of a pattern recognition technique is to represent each element as a pattern and to assign the pattern to the class that best describes it. They describe the stages in a pattern recognition problem as: Data collection, Feature selection or feature extraction, Definition of patterns and classes, Definition and application of the discrimination procedure, Assessment and interpretation of results. They also say that there are two approaches in statistical pattern recognition. The supervised approach, in which the information about how to group the data into classes is known before examining the data, and the unsupervised approach, in which no prior information about the data is known.

The authors then continue with presenting their system, Tunnel Hunter. Tunnel Hunter aims at detecting tunneling activities over the HTTP and SSH protocols, focuses on building an accurate description of legitimate traffic and builds on known pattern recognition techniques. The procedure of building patterns and classes includes gathering the features directly from the legitimate flows composing the TCP session, with each TCP flow represented by a pattern which takes into account the packet size s_i, the inter-arrival time Δt between two consecutive packets and the number of packets r that are useful for measurement .
The authors also present the issue of how many classes one has to consider when creating a classifier. They propose two approaches to the issue. One is to train the classifier with flows from a single target class (one-class classifier) and the other is to add new classes composed of outlier flows to the analysis (multi-class classifier). In the case of the one-class classifier, only one class is used, which represents the legitimate flows. The system compares the flows with this one class and decides if the

flow is legitimate or not. In the case of the multi-class classifier, the runs go through three states: State A starts with just one class, which represents the legitimate flows, just like the one-class classifier. In State B, an outlier class is created which represents some of the non-legitimate flows with specific features. This helps recognizing some non-legitimate flows which look like legitimate and would be classified as legitimate by the one-class classifier. In State C, the outlier class expands to more specific classes and this helps even more the recognition of the non-legitimate flows.

The authors then present the experimental results for the two different techniques. They used a large amount of legitimate TCP flows for training the classifier and then tested it with a large amount of tunneling flows and also a large amount of legitimate flows in order to see if the classifier can block the tunneling flows and let the legitimate flows to pass.
For the one-class classifier they present experimental results for both HTTP and SSH. Their results were very high, mostly for HTTP, for which the hit ratio of the legitimate flows was 98.71% which means that the system let the 98.71% of the legitimate flows to pass, and for the non-legitimate (tunneling) flows the results were: 100% for POP3, 100% for SMTP, 100% for CHAT and 88.09% for P2P. The results for the tunneling flows represent the percentage of the flows that were blocked by the system. In the case of SSH the corresponding percentages were: 99% for the legitimate flows of SSH, 99.1% for the legitimate flows of SCP, 88.55% for POP3, 99.92% for SMTP, 90.69% for CHAT and 82.45% for P2P.
For the multi-class classifier they present experimental results only for SSH. The results were higher than the ones for the one-class classifier. During State A the results were just the same with the results of the one-class classifier because there is only one class used. During State B, the results change for the non-legitimate flows but remain the same for the legitimate flows as such: 99% for the legitimate flows of SSH, 99.1% for the legitimate flows of SCP, 99.34% for POP3, 99.92% for SMTP, 99.92% for CHAT and 99.53% for P2P. During State C, the results become even better for the non-legitimate flows and the legitimate flows of SSH but there is a slight reduction in the case of legitimate flows of SCP as such: 99.33% for the legitimate flows of SSH, 98.91% for the legitimate flows of SCP, 99.72% for POP3, 100% for SMTP, 100% for CHAT and 99.79% for P2P.

The authors also discuss some potential attacks on the Tunnel Hunter. One case is if an SSH tunnel is initially used for remote administration and then for tunneling other protocols. In this case the first state is legitimate and the classifier will label the session as authorized. The other case has to do with packet-size and timing value manipulation. Someone can change the packet size in a flow and the time between the packets in order to disguise even further the flows to look like specific protocols, like HTTP or SSH.

In conclusion, we see that Tunnel Hunter can recognize whenever a generic application protocol is tunneled on top of HTTP or SSH with a very high rate of success, judging from the very promising experimental results. We also see that increasing the knowledge of the system can significantly improve its performance. Another important result is that no packet is excluded from the search, except from some packets during the SSH authentication phase, which are irrelevant anyway because they cannot be used for tunneling. Therefore the system has completeness near 100% (exactly 100% for HTTP).