**PCAP Parser Project by Kazi Ahsan and Jayne Rutledge Date: 1/23/2021**

We are the team behind pcap parser. We are using both bash and python scripting to run our program for automation pcap analysis.

Many times, as a security threat analyst, we are asked to investigate suspicious activity generated from the network traffic that has come into our environment. Our job is to hunt them down, detect malicious activity, contain the threat and then eradicate it. Our focus is on understanding how they penetrated our security parameters, and what hostile activity they are attempting to get away with; find out what damage has been done; and if the intruder left anything such as a new user account, a trojan horse or perhaps some new type of worm or bot software behind. Many times, users unknowingly download a piece of malware, unaware of their consequences. These users didn’t realize they’re in danger of compromising business operations or personal information.

Viruses can be spread through email and text message attachments, Internet file downloads, and social media scam links, and viruses can hide disguised as attachments of socially shareable content such as funny images, greeting cards, or audio and video files.

A computer virus attack can produce a variety of symptoms from frequent pop-up windows, make changes to your or a criminal taking control of your account or send emails in your name from another infected computer, frequent crashes causing your device to freeze or crash, unusually slow computer performance, unknown programs that startup when you turn on your computer, or unusual activities like password changes which could prevent you from logging into your computer.

Oftentimes, we never find out who the intruder actually is. As a threat researcher, from the start we know that we aren’t receiving all of the traffic information. This is possibly due to the servers’ default gateway or routers blocking rules. We haven’t been given all the pertaining logs of events which occurred simultaneously, or it may be that our email filters have blocked 99% of malicious emails that are coming in before it gets across the wire.

Our job is to look beyond the obvious, analyze network traffic and understand its behavior to find the answers.

Wireshark is a packet analysis tool that can be used in identifying and categorizing various types of attack signatures. The captured packets can reveal the signatures of attacks and identify some vulnerabilities but it doesn’t have a way to identify if a virus may be hidden in a picture or in an application. This is the reason our team’s software was created, because it can identify objects within the files, sent through http that may be malicious. We believe that these tools together can aid in a faster recovery and enable us to recover the systems from damages caused by the attackers.

To illustrate this scenario, the information given to us is that someone believes they might have downloaded malware onto their desktop. We want to find the infected downloaded files and their signature hashes.

Our tool of choice is to use Wireshark to help guide us in identifying the malware and identify where they came from. We are using a pcap file from malware-analysis-traffic.net to generate network traffic for this scenario.

The first thing we will do is go up to Statistics, to Protocol Hierarchy to see what protocols are being used in this pcap. We are interested in TCP application traffic. We can see that there is a lot of HTTP which is related to web traffic. We will select a filter to return only on layer 7, HTTP protocol. For this demonstration, we are interested in retrieving GET and POST requests for now. So we type in http.request. To get the actual file we need to go to File, Export Objects, and HTTP. We can now see all the file objects that were downloaded in this packet capture. We then sort by Content Type. In this save we see gifs, and text and applications. In this example, there are three different types of Applications which appear suspicious we will look at: java, Microsoft executable download and shockwave-flash. I’m going to click on the file and save one of each type of Application Content.

Now, this is where our blue team tool PCAP Parser comes in. Hashing is a common method used to verify the authenticity of a file and an excellent way to identify known viruses. Our program utilizes Message-Digest Algorithm 5 (MD5) hash function which is one of the most commonly used for malware analysis. Right now our program is going to check the hashes for any known virus. By verifying the hashed signature to be malware we will then know the appropriate corrective actions to take.

When we are parsing pcap data into our software, our software captures the packet data, sends it to a file that is the same as the pcap name. Then it converts data into hashes and compares them with some known malicious malware hashes that already exist in our software. When malicious software is run through our hashing program it produces a unique hash that identifies that malware (a sort of fingerprint). Once detected, the hash (es), malware, the infected file name, and the program run-date will be exported and appended to a .csv file. This file will hold all the records. We can use this file (weekly/monthly) to analyze the data, and find out which pcap files are mostly infected, what kind of malware is attacking the most or how frequently we are getting a malware match.

With this process we can check if our pcap file is infected with malicious malware or not. PCAP files which is used for forensic evidence or malware research or investigation can use this software.

As you can see knowing the hash values are vital in aiding security researchers, SOC teams, malware hunters, and reverse engineers. With this information we have now obtained, we can isolate the infected computers, and block the sites that sent the malicious malware.

A simplified process of developing a program to test software, automation testing reduces human effort. An automation testing program or script uses test data. The data is automatically entered to generate output. This further creates an analysis of the entire data, measuring the efficiency of the program.