Computer Networks - Fall 2018
Protocol Dissector Development System

Project Background

It is critical to ensure the survivability and lethality of tactical technology systems used by the warfighter. Security analysts aim to ensure this by effectively and continuously testing and evaluating these systems from a cybersecurity perspective before they are released to the warfighter in the field. Many of these systems are networked in some manner and the network communications between them are an important part of the analysis.

Project Contacts

Andres Cuevas - SLAD - andres.g.cuevas3.civ@mail.mil
John Rivers -SLAD - john.a.rivers16.civ@mail.mil

Project Description

Network systems communicate by using a layered architecture to pass information among nodes. Data at each layer must follow a protocol specification so that nodes involved will be able to send, receive, and interpret information. A protocol implementation is software that executes on communicating nodes and adheres to the protocol specification. Public protocol specifications are published and made available as Request for Comments (RFC) documents (e.g., see https://www.ietf.org/rfc.html).

A widely used and well known tool that analyzes network protocols and communication is Wireshark (Home URL: https://www.wireshark.org/). Wireshark allows us to view network communication data, or packets, in a human readable format. As part of our test and evaluation, we use Wireshark to analyze network data and assess the security level and weaknesses of the network protocols being used. Many of the protocols used are very common and follow pre-defined protocol definitions and specifications. Using the defined specification, formatted files called dissectors are written that detail the format of the layers particular to the specified protocol. Once dissector files are defined for a protocol, they can also be referenced in future dissector files when needed. As Wireshark receives network packets, it will call the necessary dissector files to parse the data out accordingly.

With security concerns on the rise, more and more custom network protocols are being developed as an active response. Many times, custom or non-IP protocols lack parsers (including Wireshark dissectors) and, therefore, analysts are not able to read and display the communication efficiently.

In some cases, protocol specifications are available and analysts may write a conforming dissector. For example, Wireshark dissectors may be written in C (native) or as Lua scripts. This is a time consuming process that requires extensive knowledge of the underlying dissector platform, the application programming interfaces (APIs), and once written is not platform-dependent. When the specification is not available, the dissector implementation process is much more difficult as analysts must infer the protocol behavior through extensive trial and error.

The Dissector Generator system will alleviate the issues described above by allowing analysts to create protocol dissectors through a graphical workspace.

Project Requirements

1. Create a standardized file format for defining a protocol structure (e.g., XML, JSON)
    a. Protocol configuration includes layers, dependencies with other protocols (e.g., HTTP uses TCP which uses IP), and data that will be visible in Wireshark (e.g., expert info, checksum)
    b. Protocols are identified by field values (e.g., with TCP/UDP packets this is commonly done with ports – tcp.port 80 indicates the HTTP dissector should be used), which may exist in field values in other protocol layers.
2. Develop an algorithm to convert the standardized file format into a Lua protocol dissector for Wireshark
3. Create a proof of concept:
    a. Input a protocol specifications (ICMP/v4, RIPv1, and RTP)
    b. Generate Lua script with implemented specifications
    c. Show that the generated Lua script correctly dissects the appropriate packet in Wireshark
        i. Import the Lua script into Wireshark
        ii. Load the appropriate pcap file for that protocol
        iii. Verify that Wireshark has correctly parsed the packet
    d. Proof of concept must be written in C or C++

There are several publicly available resources related to Wireshark dissectors, Lua, and examples. These are a few that will help:

- https://wiki.wireshark.org/Lua/ -- high-level description of Wireshark dissectors written in Lua
- https://wiki.wireshark.org/Lua/Dissectors -- short descriptions and examples of Lua dissectors
- https://wiki.wireshark.org/Lua/Examples -- extensively annotated Lua samples for UDP DNS and TCP (with reassembly) dissectors; we strongly suggest that you carefully analyze and tinker with each of these
- https://mika-s.github.io/wireshark/lua/dissector/2017/11/04/creating-a-wireshark-dissector-in-lua-1.html -- detailed tutorial about how to create a Lua protocol dissector for Wireshark