Network Traffic Proxy System

Project Background

Technology system security is critical in the world we live in today. Security analysts are tasked to continuously test and evaluate network systems from a cybersecurity perspective. An important aspect that must be tested is the way that systems communicate, including the protocols and backend processing software components.

Project Contacts

Dr. Jaime Acosta – Research, Development and Engineering Command [jaime.c.acosta.civ@mail.mil](mailto:jaime.c.acosta.civ@mail.mil)

Mr. Christian Murga – University of Texas at El Paso

[camurga@](mailto:camurga@)miners.utep.edu

Mr. Caesar Zapata – Research, Development and Engineering Command

[caesar.zapata.civ@mail.mil](mailto:caesar.zapata.civ@mail.mil)

See <http://www.icir.org/vern/cs161-sp13/notes/Networking-terminology.txt> for terminology definitions.

Project Description

Network systems communicate 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>). These documents specify the specific structure of messages (the transmission units sent between communicators), fields (the data pieces that make up messages), and the state transitions for the protocol. Network sniffers, such as Wireshark and tshark, are tools that are commonly used for observing network traffic.

Scapy (see https://scapy.net/) is an interactive packet manipulation program that is capable of reading and decoding packets, modifying values, and creating packets, among many other functions. Scapy can be run as a Python module or as stand-alone and supports Python2 and Python3. NFQUEUE is a Linux technology that allows intercepting packets at the kernel level (before they reach operating system services). See <https://home.regit.org/netfilter-en/using-nfqueue-and-libnetfilter_queue/> for an example of C and NFQUEUE used together and <https://gist.github.com/eXenon/85a3eab09fefbb3bee5d> for an example of scapy and NFQUEUE used together).

During security testing it is critical to identify any weaknesses that may exists in the communication services. This is commonly done by using publicly available software that adheres to the protocol used by the communication service. However, required is a system that is capable of intercepting packets and allowing an analyst to interactively modify packets on the fly.

As an example, assume that an analyst is testing the Internet Control Message Protocol (ICMP). Ping is a program that works by using the ICMP protocol. Most operating systems have a ping program pre-installed and can be run from a terminal (see <https://en.wikipedia.org/wiki/Ping_(networking_utility)> for an example). To test ICMP using ping, an analyst could use this system to intercept ping packets (either from the local or another remote machine) and then modify values (e.g., unintended, larger than expected) to test the validation mechanism of the target device’s ICMP decoding and processing software.

This proxy behavior used in testing is not new, but lacking in current systems is an integrated, graphical environment that supports protocols for multiple network layers. For example, BurpSuite is a software system that currently performs a similar task, except that it only supports the Hypertext Transfer Protocol (HTTP), which an application-layer protocol.

Scapy Proxy Requirements

Traffic

The Network Traffic Proxy System will have several functionalities that will be made available to analysts from a graphical interface. The system should allow analysts to set iptable rules (required for proxy behavior) and save these configurations. Analysts may specify a filter that will determine which packets will be intercepted (e.g., if TCP and port 80 are specific only packets with a TCP layer and port 80 will be intercepted). While packet interception is enabled, the analysts should be able to modify, forward, and/or drop selected packets. The packet contents (fields and their values) should have several display modes including raw hexadecimal and decoded-type (i.e., the way that scapy dissects or decodes the field values). These values will be editable by the analyst. In addition, the packet data should be separated by layer (e.g., Ether, IP, TCP, ICMP). The system needs to keep track of intercepted packets in a queue and allow the analyst to select any packet in the queue.

Packet Capture Files (PCAPs)

Packet capture files are the most commonly used format for storing network traffic. This system should save all intercepted traffic to a PCAP file. In addition, the system should be able to load a PCAP file and display it’s packets for an analyst to modify and send regardless of whether live packet interception is on or off.

Hooks

Hooks are functions that execute code snippets and are run automatically when some conditions are met. For example, a hook may be written that modifies the source IP address of all packets that have a destination port 8080. The analyst should be able to load hooks and enable/disable them. Several hooks may be combined into a hook collection that can also be loaded/enabled/disabled. Analysts should be able to also manage hooks (view status and enable/disable).

Fuzzer

A fuzzer is software that generates unintended or otherwise random inputs to a system. In particular, this system should be able to use the American Fuzzy Lop (AFL) to fuzz packets. As part of this task, an analyst should be able to choose fields of a dissected packet that will be fuzzed, use AFL to generate the packets and then enable/disable sending the fuzzed packets.

Other Expectations

The system should be usable by an intermediate to expert level analyst. The system should support queues of at least 100 packets without any behavioral performance issues on a modern laptop system. The source code should be well-designed and documented and facilitate scalability and maintainability.