HTTPVIZ: A Tool for Visually Inspecting Real-time HTTP Data

Sean Kelley (sean.kelley@tufts.edu)

Ming Chow (mchow@cs.tufts.edu)

Abstract

Analyzing network traffic is an inherently large-scale operation. Those inspecting traffic on a network cannot simply check every packet that they receive; they must have automated assistance to simplify the raw data into meaningful chunks. Unfortunately, network traffic is so diverse that providing detailed domain-specific information for traffic is difficult and requires some specialization towards a given type of traffic (i.e., protocol). With users' increasing reliance on websites to do what once only desktop applications did, HTTP has become a protocol worth specializing in. This project aims to provide network traffic analyzers a tool to visually inspect the content of HTTP traffic on a network, either in real time or retroactively, by providing an interactive layout and display of the all files being transferred over the network. The tool will also provide searching and filter functions to allow users to concentrate their efforts on particular subsets of interesting traffic.

Introduction

Administering a large network, in particular private networks where content is to be filtered and access to different types of content controlled, is time-consuming and difficult to perform effectively in both everyday and emergency scenarios. Needless to say, active networks produce enormous amounts of data. Over the years many automated tools have been built to assist network administrators, allowing both real-time and retroactive analysis of complete packet data. One such tool is the well-known Wireshark, which in addition to collecting packet data and providing powerful filtering and search capabilities, allows the reconstruction of unencrypted HTTP data.

HTTP data is a particularly rich set of information for network analyzers because of its combination of widespread use and semantically-dense, mostly-human-readable content. A small several-kilobyte HTML page can provide significantly more information on a network's content or person's usage habits than a single large media file, for instance. Many services, both with and without personally sensitive information, are being created on or moved to the web and are often built on HTTP to take advantages of users' familiarity with browser-based interfaces. In short, analyzing HTTP is an extraordinarily efficient way to extract meaningful information on the content of a network, as opposed to simply collecting and analyzing usage statistics.

General-use HTTP data (i.e. not that used for sensitive personal information such as social security numbers) is still unencrypted, for the most part. Even some sites that may carry such information, such as Facebook, are still unencrypted (this is something the reader can test themselves -- if HTTPS has not been explicitly enabled, Facebook will not use it). Despite the widespread support for the encrypted HTTPS protocol, many sites remain on unencrypted HTTP for the sake of convenience, laziness, cost, legacy support, or some combination of the four.

Analyzing unencrypted HTTP is easy with tools like Wireshark, and because of the continued lack of usage of encryption, it is likely to stay that way for some years to come.

The intent of this tool is to augment existing network analysis processes by providing network analyzers with a tool for visually inspecting HTTP data.

To the Community

Having established the information-rich nature HTTP, it is important to note what this information can be used for. The current tool is intended to perform as an extension to existing tools and tool suites and to simply speed the processes that are already established for network analysis. Often, network analysis tools provide little by way of summarizing what network users are actually seeing and opt instead of move up a level, providing huge amounts of metadata for analyzers to sift through instead. The current tool tries to fill this gap by extracting the most immediately salient information from a network -- that is, sites and media sent over HTTP -- and supplying the network analyzer with a way to identify potentially anomalous content quickly.

In this way the current tool doesn't intend to replace any other tools. It supplies incomplete, at-a-glance data to quickly inform its users of general usage trends but doesn't allow further inspection. It is by no means full-featured and powerful, but does fill an as-yet unaddressed niche in functionality.

Review and Current Methods

As previously mentioned, current methods of network traffic inspection are not geared towards visual inspection of contents. There are a number of network analyzers already available:

Cain and Abel, dsniff, ettercap, ngrep, tcpdump, Firesheep, and, notably, Wireshark are all widely used tools. These tools do not support effective data visualization. Visualizers, such as EtherApe, already exist, but try to make sense of all traffic on a network and the associated connections that are created, rather than the content of those connections.

At the other end of the spectrum, tools like Wireshark allow you to extract meaningful HTTP data from captured packets, but not in an interactive or visual way. The extracted data is indiscriminately mixed together without regard to time or which connection it came from, and the action is performed not in real time but after the fact, losing the critical time and visual elements.

In other words, there is not a tool in existence or at least widespread use that effectively displays to a network analyzer the content that would be most effective for inspecting the most meaningful usage content on any arbitrary network. This is a currently unsolved problem.

Method

The current tool attempts to unify two different classes of tools -- real-time packet analyzers and real-time network connection visualizers. Wireshark provides a powerful API for using its packet-capturing, -filtering and -reconstruction capabilities via the registering of Lua (a scripting language) functions upon starting Wireshark. The current tool was implemented by a Lua script bridging the gap between Wireshark and the Prefuse visualization library, written in Java.

HTTP is a stateless protocol, meaning that after data has been transferred the active connection is closed until another request is sent. A user of this tool cares about the semantic content of the connections, not implementation details such as the repeated establishing of connections. To this end, and for the purposes of this paper, connections between any two hosts

are treated as a single conversation.

All the existing (active or not) conversations appear in a list. Each list item doubles as one bar in a bar chart, where the horizontal length of a bar represents the total number of bytes transferred relative to the most active connection. The list can be sorted according to the user's preference (for example, by the time of the last activity, or by the total bytes transferred). List items are updated and sorted in real time as more HTTP data is given to the visualization.

Any time the visualizer identifies an image in the reconstructed packets, it renders a thumbnail of the image and displays that thumbnail moving from the server to the client. Currently, the visualizer only supports extracting a few of the most common image types -- JPEG, GIF, and PNG. Clients that elicit a large number of image responses from the server will thus be immediately obvious to the user who can them zoom and focus on the conversation in question. Many features that exist in Wireshark can be used to affect the visualization. For example, the ability to filter HTTP packets by content can be reflecting the visualization, if so desired.

Acting merely as a monitor to the network content, the tool is light on user interactions. Additionally, this version of the tool is merely a prototype, and there a number of improvements to both the feature set and usability that will be discussed in the future work section.

Summary

Despite being only a prototype, the current version of the tool shows promise. In some informal tests of its abilities, it was able to make certain behaviors that were not obvious in other analysis tools (Wireshark, Etterape), such as particular users consuming most of the bandwidth over a particular period. Formal user studies were not conducted (and will not be, for some time)

due to the lack of features and stability.

The ultimate goal is to create a tool extension that will become widely used among network analyzers. To that end, development will continue. If nothing else, it is hoped that tools like these will encourage internet users to accelerate their migration to total usage of HTTPS instead of HTTP. As witnessed during the iPhone-location-tracking fiasco, in which Apple was discovered to be storing timestamped location data on iPhones, average users don't respond well to potential security threats until their implications are made directly visible. Perhaps even more startling was the response to the release of Firesheep, when it had a meteoric rise to fame for putting traffic sniffing and some exploits in the hands of everyday users. Giving network analyzers the ability to visually inspect network users' data in parallel and real-time may be the push that the public needs to understand the problems with not using HTTPS on sites that store personal information, such as Facebook.

Future Work

Aside form continued development of existing features, the prototypic nature of the current incarnation means there is a large pool of feature additions to choose from. There are two features in particular that would be very useful to have. The first is the ability to have a history of the salient data kept and allowing the user to scrub back and forth through history using a slider. There are implementation difficulties with respect to memory usage and tracking previous states, but the usefulness of being able to identify anomalous activity and then move backwards in history to examine the conditions that led to the activity cannot be overstated.

The second major feature would be to support more than just images. Complex media such

as videos would be difficult to support, but even attaching an HTML renderer and showing images in the context of web page thumbnail is a big step up in functionality. Rather than seeing a collection of out-of-context images, network analyzers would be able to see something very similar to what the network users are seeing, including the ever-important context.

There are a number of lesser features that would be worth including but would have less of an impact on the capabilities of the visualizer. These features include allowing user-selected visual properties such as changing the meaning of the bar chart or coloring conversations based on user-specified criteria, and complementing the history scrubber with an optional sliding window in order to visualize only the most recent and interesting data.

This tool or tools like it have the potential to become an important component in real-time analysis, but much work needs to be done to bring their feature set to a position in which the tool is able to speak for itself in terms of usefulness.

References

Butler, Eric. "Firesheep, a day later." codebutler. 26 Oct 2010. Web. 12 May 2011. <http://codebutler.com/firesheep-a-day-later>.

"EtherApe: Introduction." EtherApe. SourceForge, n.d. Web. 13 May 2011. <http://etherape.sourceforge.net/introduction.html>.

Goodin, Dan. "No, iPhone location tracking isn't harmless and here's why." Register 22 Apr. 2011: 1-2. Web. 12 May 2011. <http://www.theregister.co.uk/2011/04/22/apple_iphone_location_tracking_analysis/>.

Heer, Jeffery, Stuart Card, and James Landay. "prefuse: a toolkit for interactive information visualization." Proceedings of the SIGCHI conference on Human factors in computing systems. Portland, OR, 2005. PDF.

Lamping, Ulf. "Wireshark Developer's Guide." Wireshark. Wireshark Foundation, 2010. Web. 13 May 2011. <http://www.wireshark.org/docs/wsdg_html_chunked/>.