Welcome to curl-loader

Table of contents

1. Introduction

curl-loader (also known as "omes-nik" and "davilka") is an open-source tool written in C-language, simulating application load and application behavior of thousands and tens of thousand HTTP/HTTPS and FTP/FTPS clients, each with its own source IP-address. In contrast to other tools curl-loader is using real C-written client protocol stacks, namely, HTTP and FTP stacks of libcurl and TLS/SSL of openssl, and simulates user behavior with support for login and authentication flavors.

The goal of the project is to deliver a powerful and flexible open-source testing solution as a real alternative to Spirent Avalanche and IXIA IxLoad.

The tool is useful for performance loading of various application services, for testing web and ftp servers and traffic generation. Activities of each virtual client are logged and collected statistics includes information about resolving, connection establishment, sending of requests, receiving responses, headers and data received/sent, errors from network, TLS/SSL and application (HTTP, FTP) level events and errors.

Virtual clients are grouped together to the so-called batches of clients, performing the same sort of activities, like:

- authentication login;

- user activity simulation by fetching several URLs with configurable timeouts in between;

- logoff.

The tool can be easily extended to generate sftp, telnet, tftp, ldap, ssh, scp etc other application protocols, supported by the great libcurl library.

2. Features List

- Virtual clients number. The tool runs, depending on your HW and scenario, 2500-100 000 and more simultaneously loading clients, all from a single curl-loader process. Actual number of clients may be several times higher, and it is limited mainly by memory. Each client loads from its "personal" source IP-address, from the "common" IP-address shared by all clients, or from the IP-addresses shared by some clients, where a limited set of shared IP-addresses can be used by a batch of clients.

- Rampup of the virtual clients number at loading start in either automatic or manual mode;

- IPv4 and IPv6 addresses and URIs;

- HTTP 1.1. GET, POST and PUT, including file upload operations;

- HTTP user authentication login with POST or GET+POST methods. Unique configurable username and password for each virtual client as well as configurable posted string (post-forms) are the options. Another option is loading of users with credentials from a tokens text file;

- HTTP POST/GET forms with up to 16 tokens filled from a tokens text file;

- HTTP user logoff with POST, GET+POST, or GET (cookies); POST logoff with configurable posted string (post-forms);

- HTTP multipart form data POST-ing as in RFC1867;

- HTTP Web and Proxy Authentication (HTTP 401 and 407 responses) with Basic, Digest (RFC2617) and NTLM supported;

- HTTP 3xx redirections with unlimited number of redirections;

- HTTP cookies and DNS caches;

- FTP passive and active, FTP upload;

- Full customization of client request HTTP/FTP headers ;

- Transfer limit rate for each client download or upload operation on a per url bases;

- URL fetching probability;

- TCP connections reuse or re-establishment on a per url bases;

- Unlimited configurable number of URLs. Mixing of HTTP, HTTPS, FTP and FTPS urls in a single batch (test plan) configuration;

- Connection establishment timers for each URL;

- URL completion timers monitoring and enforcement for each client;

- Inter/after URL "sleeping" timers, including random timers taken from a configurable interval;

- Logfile with tracing activities for each virtual client. The logfile is automatically rewinded, when reaching configurable size preventing disk crashes;

- Logging of responses (headers and bodies) to files.

- Pre-cooked batch configuration (test plan) examples;

- Load Status GUI at console and with output to file;

- Status and statistics for each virtual client, which are logged to file;

Here is a screenshot:

curl-loader screenshot

3. License

curl-loader is licensed under GPLv2.

4. Authors

The tool is written and supported by Robert Iakobashvili and Michael Moser, both from Israel.

Israel flag

Please, use the link to the mailing list provided in Support section to contact us and to get support for the tool. Subscribe to the curl-loader-devel list and mail your filled PROBLEM-REPORTING form located in the curl-loader tarball to the list. Your suggestions, ideas and patches would be very much appreciated.

The official language of the curl-loader-devel mailing list is English, whereas Russian and German written mails are also welcomed.

5. Successfully Used

To simulate HTTP/S load of thousands of clients against authentication gateway for testing of the gateway performance in various scenarios. curl-loader supplied HTTP/S client load against Apache web-server with the gateway in the middle, where the gateway made a browser hijacking and HTTP- redirection of the curl-clients to the HTTPS url at the gateway's own web-server. HTTPS page of the web-server provided a POST form to the user with username and password for the client/user authentication against an external AAA (RADIUS) server. If the authentication was OK, user (a libcurl virtual client object) was allowed to enter the Internet and to perform some sort of simulated by curl-loader network activity, namely, fetching urls and sleeping in between them. After enjoying Internet, user was coming to logoff.

To test web-server pages, authenticating tens and hundred thousand of clients, where each client comes to a HTTPS url using GET method and is redirected by the web-server to another url, providing authentication POST form with username and password. After successful authentication of a client the web-server was setting to the client server-cookies. Client activities were further simulated by fetching urls and sleeping in between. Clients

were doing logoff using GET-method to the web-server logoff-url, where the cookies were used by the web-server to verify client identity.

To generate Gbps traffic from thousands of TCP/HTTP clients and to test the impact of thousands of firewalling and NAT iptables/ipset rules and hundreds of the rules being added/deleted each second at performance of a gateway device. curl-loader provided client load against Apache web-server fetching a url with a gigabyte file, thus, creating a permanent heavy-load traffic, containing thousands of tcp-streams at the gateway in the middle.