Red7 Communications, Inc.
PO Box 27591
San Francisco CA 94127-0591
USA

Online at:
www.cyberspark.net

To: CyberSpark Open Source Project

From: Sky

Date: October, 2014

Subject: Documentation of **CyberSpark** analysis data and software.

This document: was designed to be usable in OpenOffice, LibreOffice, Microsoft Word and other software capable of using these formats. Only the DOC version is made available online, and we check it to be sure it's at least readable in the other programs.

Table of Contents

I would like to thank John D. Tangney, who has assisted in code reviews and added some good ideas, and reflection, to the project. And the code is in use at other installations, as well as at my CyberSpark project, so I know it's possible to make it work outside the limited testbed. —"Sky"

Overview

The CyberSpark sniffers record their progress in log files. These are saved locally and then can be offloaded for analysis. The sniffers are designed to be aloof; do not expose any services; and generally are not in DNS. The *Analysis* system adds the possibility of coordinating log data between sniffers and a central location, in batches or in real time, to a database where logged events can be stored. This data can then be interrogated, and analyzed or displayed. The data can be analyzed and displayed retrospectively or in real time. We provide some graphing using the D3js package.

The Data (logs etc.)

The logs all contain these fields:

milliseconds: Unix millisecond time for the log entry. This is “Unix time” and is in UTC.

date: Unix date for the log entry. This is a human-readable date, and is in local server time with an indication of offset from UTC.

host: A host designation such as cs9[50.56.216.34]". This is free-form data of any length, but we truncate it to 40 characters for the database.

thread: A thread designation such as CS9-0. This is also free-form, and we truncate to 40.

tick: Number of iterations of the sniffer daemon since it started.

crashes: Number of consecutive crashes (loops) of the daemon.

http_ms: Seconds for the HTTP GET or POST to return a result. Note this is not milliseconds even though that’s what we originally wanted it to be.

length: Length of the returned HTTP response.

md5: An MD5 hash stored by certain filters.

condition: The names of the filters that were applied. For example “dns,ssl” The database column is named “conditions” plural.

url: The URL that was analyzed. Note this is not stored in the database directly for each log entry but instead we store URLs in a separate table and an index ("ID") within the main logs tables.

result_code: HTTP result code of the GET or POST.

year: Year as a number.

month: Month as a number.

day: Day of the month as a number.

hour: Hour of the day as a number.

minute: Minute of the hour as a number.

second: Second of the minute.

APIusage: Number of times the external GSB interface was hit (if "gsb" is being used).

full_message: A full descriptive message, as returned by each and all filters.

Sample data:

```
1323825802666,"Tue, 13 Dec 2011 17:23:22 -0800","cs9[50.56.216.34]","CS9-
0",8,0,0,2066,,"dns,ssl","https://base.red7.com/",200,2011,12,13,17,23,22
,0,"OK        https://base.red7.com/ OK        [basic]   OK        [dns]
OK        [ssl] Verify SSL/HTTPS on base.red7.com     "
```

Many log entries correspond to local events, such as server start-up or shut-down, email errors, and delays during sniffer operation. These can generally be detected because they are missing a *result_code* value or missing a *url*.

Analysis Database

The system is designed to run on Linux, and more specifically was developed on Ubuntu 8.04 LTS through Ubuntu 14.04 LTS. Since it's basically in PHP, it should run on other operating systems with minor modifications. The database software is MySQL 5.5.37 (as of 2014-06-26). We're using PHP 5, and specifically 5.3.10 as of this writing.

We make the assumption that since you're working in a sensitive area you will have your software pretty much up to date and thus we focus primarily on "recent" versions of operating system, MySQL and PHP.

The database *cyberspark_analysis* contains three tables.

logs: This table contains the actual log entries. There are on the order of tens of millions of these entries from nearly 5 years, as of mid-2014. They are indexed on ID (record number) which is unique, URL_HASH which is a hash of the URL corresponding to the particular log entry (line), and milliseconds. In general we just access log entries based on the URL we're interested in and a time span of some sort, so that's all we really need to index. In the future you might want to analyze by host or condition or HTTP *result_code*, but you can just index your tables at that time. Everything in this table is fixed length, making storage and retrieval more efficient. SQL to create these tables appears later in the document.

urls: These are the plain text URL values corresponding to the log's *URL_HASH* and *URL_*ID entries. As a log entry goes into the *logs* table, the URL it's connected with gets looked up in *urls* and if it's new, a new *urls* entry is created. In any event, the URL_ID goes into the *logs* entry so we can know the actual URL if we need to. In general we are analyzing only a single URL so we don't look this up anywhere as often as we look up and retrieve log entries. We use *URL_HASH* more often because it is a fixed (32-character) length which is more efficient for MySQL.

messages: These are the messages returned within log entries. Because these are variable length text, and are in general not searched nor analyzed nor graphed, we keep them in this full text table. Each line in the table corresponds to a line in the *logs* table. The record ID numbers match between the two tables. Most log entries have a message, even if all it says is that everything's OK.

Log Transport (from sniffers to the analysis system)

We have two PHP scripts we use to create database entries. The first one takes a log file and reads it line by line, uploading log data to the database. It ignores duplicate data (determined by the *milliseconds* time combined with *URL_HASH)*. It is designed to be run from a URL. The second script can also be hit at a URL and takes POST data corresponding to a single log entry. An API key is also required to be present in the POST data…there is no other authentication.

When processing a full log file, upload it to the analysis server and make sure it's unzipped to its original native text format. Log files are gzipped when they are rotated by a sniffer. A whole directory of log files can be processed in one swoop. Put them into a directory, unzipped. Go to the URL /analysis/cs-log-file-select.php and specify the path (on the server) to the log, or to the directory containing logs. They will be processed.

Individual log entries can be transmitted to the analysis system in near real time from any sniffer. To enable this capability, put the file /log-transport.php and its config and pw files in the /usr/local/cyberspark/ directory on the sniffer server. When you launch CyberSpark Service again, a log-transport.php process will be started by each cyberspark.php. If log-transport-pw.php contains a **CS_API_KEY** and if cs-log-pw.php contains that same CS_API_KEY, then the API_KEY parameter must be present (and the two files' values must match) in all requests that upload data.

In addition to requiring a CS_API_KEY, you may configure your log analysis and processing within a directory protected by HTTP "Basic Authentication." This requires a user name and a password to access either the analysis or the PHP scripts that do the processing. Both the processing and the analysis can be done from the same base location on your web server. We use /analysis/ for our operations. The HTTP Basic Authentication name and password must be included in two files:

On the analysis web server, the HTTP user+password and the CS_API_KEY are defined in this file:
/var/www/analysis/cs-log-pw.php

On each sniffer that's going to send data in real time the same items are defined in:
/usr/local/cyberspark/log-transport-pw.php

The values in the two files must match each other.

The cs-log-pw.php file also contains MySQL authentication information, which is required on your analysis server. The sniffers do not need this information, as they do not use MySQL.

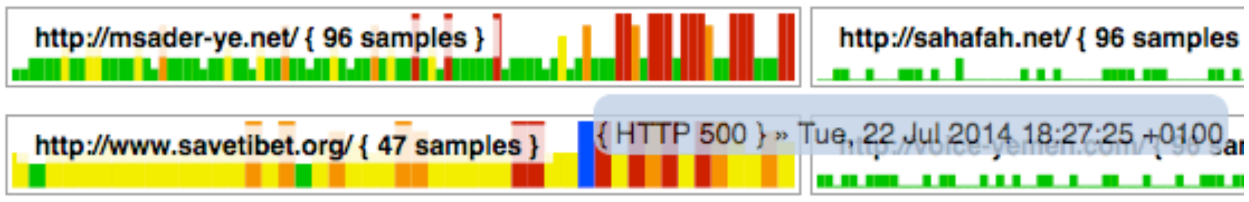
If you do not need real-time data transport, you may omit /log-transport.php from the sniffers, restart CyberSPark, and then just transfer the log files to another location for analysis.

To restart CyberSpark on a sniffer:
service cyberspark start

Analysis and Beautiful Graphics

We use the D3js package to fetch and display the data. (http://d3js.org/) Data can be fetched from historical data or real-time data, and in both cases comes from the database, which of course must be updated in real time in order to provide real time graphing and display. We built a special type of chart in D3 that indicates "server hotness" by color, and has mouseover "tooltips" that show date and server conditions. These strip charts are designed to occupy minimal vertical space so they can be stacked on a compact page. They'll operate on desktop or on touch-screen browsers.

A modified *horizon chart* has been designed. This chart combines color, position and a rollover to present several types of information at once for each URL that is analyzed. The colors green, yellow, orange and red indicate the relative response time of the site at a given time. (Green»0,1; yellow»2,3; orange»4,5; red»6+) Blue bars indicate HTTP failure codes, such as 500 (illustrated below). Magenta bars indicate timeouts (usually 60+ seconds).



Database structure

```
-- MySQL dump 10.13  Distrib 5.5.38, for debian-linux-gnu (x86_64)
--
-- Host: localhost    Database: cyberspark_analysis
-- ------------------------------------------------------
-- Server version   5.5.38-0ubuntu0.12.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `files`
--

DROP TABLE IF EXISTS `files`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `files` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `name` text NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=2585 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `logs`
--

DROP TABLE IF EXISTS `logs`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `logs` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `URL_HASH` varchar(32) DEFAULT NULL,
  `milliseconds` bigint(13) unsigned NOT NULL,
  `date` varchar(40) NOT NULL,
  `host` varchar(40) NOT NULL,
  `thread` varchar(40) NOT NULL,
  `tick` int(7) unsigned NOT NULL DEFAULT '0',
  `crashes` int(4) unsigned NOT NULL DEFAULT '0',
  `http_ms` double unsigned NOT NULL DEFAULT '0',
  `length` int(10) NOT NULL DEFAULT '0',
  `md5` varchar(32) DEFAULT NULL,
  `condition` varchar(100) DEFAULT NULL,
  `URL_ID` int(10) NOT NULL,
  `result_code` int(3) unsigned NOT NULL DEFAULT '0',
  `year` int(4) unsigned NOT NULL,
  `month` int(2) unsigned NOT NULL,
  `day` int(2) unsigned NOT NULL,
  `hour` int(2) unsigned NOT NULL,
  `minute` int(2) unsigned NOT NULL,
  `second` int(2) unsigned NOT NULL,
  `APIusage` int(6) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  UNIQUE KEY `ID` (`ID`),
  UNIQUE KEY `NO_DUPE_LOGS` (`URL_HASH`,`milliseconds`),
  KEY `milliseconds` (`milliseconds`),
  KEY `URL_HASH` (`URL_HASH`)
) ENGINE=InnoDB AUTO_INCREMENT=6485872 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `messages`
--

DROP TABLE IF EXISTS `messages`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `messages` (
  `ID` int(10) unsigned NOT NULL,
  `message` text,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `ID` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `urls`
--

DROP TABLE IF EXISTS `urls`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `urls` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `URL_HASH` varchar(32) NOT NULL,
  `url` text,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `ID` (`ID`),
  UNIQUE KEY `URL_HASH` (`URL_HASH`)
) ENGINE=InnoDB AUTO_INCREMENT=821 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

```
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2014-10-15  3:38:49
```

The To-do list

° select an actual month at a time

?year=YYYY&month=MM

?year=YYYY&week=WW

° there can be too many points and the bars get too small to mouseover => how to resolve this and discard some data

» processing log entries if you see there are too many (SIEVE=250) the digest them using some logic

s° ooo